The **syslogd** command creates the file **/etc/syslog.pid**, containing a single line with its process ID. This file can be used to kill or reconfigure **syslogd**. To bring **syslogd** down, it should be sent a terminate signal. For example:

```
kill 'cat /etc/syslog.pid'
```

Flags

-d	Turns on debugging.
-f *configfile*	Specifies an alternate configuration file.
-m *markinterval*	Specifies the number of minutes between mark messages.

Example

To start **syslogd** daemon and and change the mark interval:

```
syslogd -m30
```

This command changes the mark interval to 30 minutes. If the configuration file contains:

```
kern,mark.notice        /usr/adm/notice
kern.err                @scott
*.info;mail.none        /usr/spool/adm/syslog
*.alert;auth.warning    darlene
```

syslogd logs kernel messages and 30-minute marks at `notice` level (or higher) in the file `/usr/adm/notice`, forwards kernel messages at `err` level (or higher) to **syslogd** on the host `scott`, logs messages at `info` level (or higher) except mail messages in the file `/usr/spool/adm/syslog`, and informs the user `darlene` of any warning message (or higher) from the authorization system.

Files

/etc/services	Contains definition of the Internet domain socket.
/etc/syslog.conf	Contains the configuration file.
/etc/syslog.pid	Contains the process id.
/dev/log	Contains AIX domain datagram log socket.

Related Information

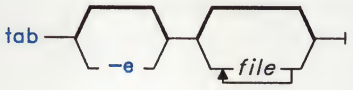
The **syslog** system call in *AIX Operating System Technical Reference.*

tab, untab

Purpose

Changes spaces into tabs or tabs into spaces.

Syntax



OL805069

OL805065

Description

The **tab** command reads *file*s (standard input by default), and replaces spaces in the input with tab characters wherever it can eliminate one or more spaces. It writes the resulting file back to *file* or, if the input was standard input, to standard output. **tab** assumes that the tab stops are set every eight columns starting with column nine.

The **untab** command reads *file*s or standard input, replaces tabs in the input with space characters and writes back to the original file or to standard output.

Flag

-e Replaces only those spaces at the beginning of a line up to the first nonspace character.

Related Information

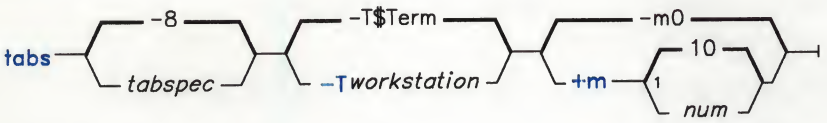
The following command: "**newform**" on page 686.

tabs

Purpose

Sets tab stops on work stations.

Syntax



[1] Do not put a space between these items.

OL805381

Description

The **tabs** command clears up to 20 previous tabs and sets up to 40 tabs on the work station according to the supplied *tabspec*. *tabspec* can be either a flag indicating an available code or column numbers. The available codes cover formats required by most structured programming languages.

When you use the **tabs** command, always see the leftmost column number as 1, even if your work station refers to it as zero (0).

If you do not specify a *tabspec*, the default value is **-8**.

Tabspecs

-a Sets the tabs to 1, 10, 16, 36, and 72 (IBM S/370 Assembler first format)

-a2 Sets the tabs to 1, 10, 16, 40, and 72 (IBM S/370 Assembler second format)

-c Sets the tabs to 1, 8, 12, 16, 20, and 55 (COBOL normal format)

-c2 Sets the tabs to 1, 6, 10, 14, and 49 (COBOL compact format, columns 1-6 omitted). With this code, the first column position corresponds to card column 7. One space gets you to column 8, and a tab reaches column 12. Files using this code should include a format specification of:

```
<:t-c2  m6  s66  d:>
```

For an explanation of format specifications, see the **fspec** file in *AIX Operating System Technical Reference.*

-c3 Sets the tabs to 1, 6, 10, 14, 18, 22, 26, 30, 34, 38, 42, 46, 50, 54, 58, 62, and 67 (COBOL compact format with more tabs than **-c2**. This is the recommended format for COBOL. Files using this code should include a format specification of:

```
<:t-c3  m6  s66  d:>
```

-f Sets the tabs to 1, 7, 11, 15, 19, and 23 (FORTRAN).

-p Sets the tabs to 1, 5, 9, 13, 17, 21, 25, 29, 33, 37, 41, 45, 49, 53, 57, and 61 (PL/I).

-s Sets the tabs to 1, 10, and 55 (SNOBOL).

-u Sets the tabs to 1, 12, 20, and 44.

In addition to the preset formats, three other types of *tabspec*s are available:

-num Sets regularly repeating tabs at every *num*th column. (**-8** is the standard AIX tab setting and the one required for use with the **nroff -h** flag.) Another special case is **-0**, which implies no tabs at all.

num[,*num*] . . . Sets tabs at the named column numbers (a comma-separated list in ascending order). You may specify up to 40 numbers. If any number except the first has a plus sign prefix, the prefixed number is added to the previous number for the next setting. Thus, the tab lists `1,10,20,30` and `1,10,+10,+10` provide the same tab settings.

--filep Reads the first line of the named *filep* for a format specification. If it finds one, it sets tabs the same way. If it does not find a format specification, it sets tabs to the system default (**-8**). Use this *tabspec* to make sure that a file has the same tab settings as those in a file already correctly formatted.

Flags

Note: If the same flag occurs more than once, only the last one takes effect.

-T*workstation* Identifies the work station so that **tabs** can set tabs and margins correctly. *workstation* is one of the work stations listed under the **greek** command. If you do not provide a **-T** flag, **tabs** uses the shell variable **$TERM**. If no *workstation* can be found, **tabs** tries a general value that works for most work stations.

+m*num* Moves all tabs to the right *num* columns, and makes column *num*1 the left margin. If **m** is given without a value, 10 is assumed. The leftmost margin on most work stations is defined by **m0**.

Related Information

The following commands: "**greek**" on page 499, "**nroff, troff**" on page 709, and "**troff**" on page 710.

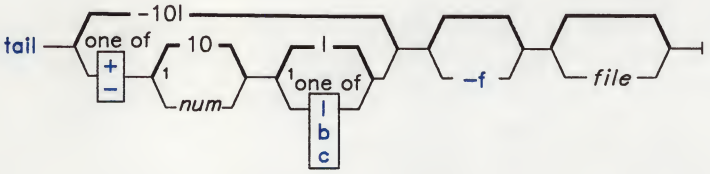
The discussion of **term** and **environ** in *AIX Operating System Technical Reference.*

tail

Purpose

Writes a file to standard output, beginning at a specified point.

Syntax



OL805303

[1] Do not put a blank between these items.

OL805308

Description

The **tail** command writes the named *file* (standard input by default) to standard output, beginning at a point you specify. It begins reading at +[*num*] lines from the beginning of *file* or -[*num*] lines from the end of *file*. The default *num* is 10. *num* is counted in units of lines, blocks, or characters, according to the subflag appearing after *num* (see the following flags).

Flags

-f Does not end after it copies the line of the input file if the input file is not read from a pipe, but enters an endless loop in which it sleeps for a second and then attempts to read and copy further records from the input file. Thus, it can be used to monitor the growth of a file being written by another process.

+[*num*]**l**
+[*num*]**b**
+[*num*]**c**
Begins reading *num* lines (**l**), blocks (**b**), or bytes (**c**) from the beginning of the input.

-[*num*]**l**
-[*num*]**b**
-[*num*]**c**

Begins reading *num* lines (**l**), blocks (**b**), or bytes (**c**) from the end of the input.

Japanese Language Support Information

+[*num*]**k** Begins reading *num* characters (**k**) from the beginning of the input.
-[*num*]**k** Begins reading *num* characters (**c**) from the end of the input.

The **c** flag in Japanese Language Support begins reading as closely as possible to the number of bytes requested, without breaking a 2-byte character. The number of characters in input containing SJIS characters may not equal the number of bytes. To get, or skip, precisely *num* characters, use the **k** flag.

Examples

1. To display the last 10 lines of a file:

```
tail  notes
```

2. To specify how far from the end to start:

```
tail  -20  notes
```

This displays the last 20 lines of `notes`.

3. To specify how far from the beginning to start:

```
tail  +200c  notes  ¦  pg
```

This displays `notes` a page at a time starting with the 200th character from the beginning.

4. To follow the growth of a file:

```
tail  -1  -f  accounts
```

This displays the last line of `accounts`. Once a second, **tail** displays any lines that have been added to the file. This continues until stopped by pressing **INTERRUPT** (**Alt-Pause**).

Related Information

The following commands: "**dd**" on page 301 and "**pg**" on page 744.

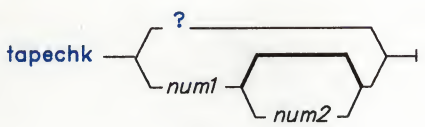
The discussion of Japanese Language Support in *Japanese Language Support User's Guide.*

tapechk

Purpose

Performs consistency checking of the streaming tape device.

Syntax



OL805445

Description

The **tapechk** command performs rudimentary consistency checking on an attached streaming tape device. Some hardware malfunctions with a streaming tape drive can be detected by simply reading a tape. **tapechk** provides a way to perform tape reads on the file level.

Since the streaming tape drive cannot backspace over physical data blocks or files, **tapechk** rewinds the tape to its starting position prior to each check. You can specify numeric arguments to control the number of files checked or skipped. If you do not specify any arguments, **tapechk** rewinds the tape and checks only the first physical block.

Note: The **backup** command allows you to archive files selectively or as an entire file system. It writes data as a continuous stream terminated by a file mark, regardless of the number of files specified. The **tapechk** command perceives each stream of data as a single file. This is important when you specify numeric arguments with the **tapechk** command.

Although you can use **tapechk** on any streaming tape cartridge, it is primarily designed for checking tapes written by the **backup** command.

Flags

num1 Checks data for the next *num1* files.

num2 Skips the next *num2* files from the beginning of the tape.

? Explains the format of the **tapechk** command.

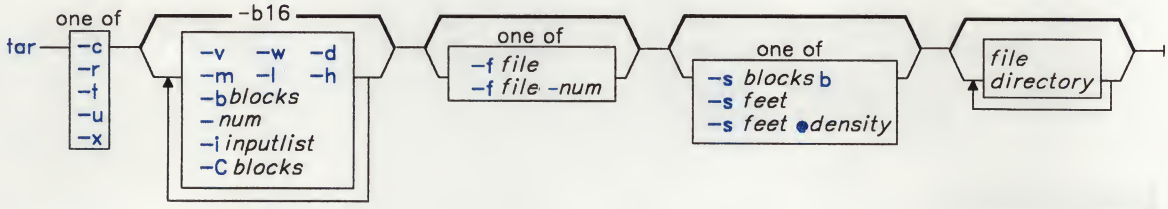
Note: If you specify this argument, it must be the first argument.

tar

Purpose

Manipulates archives.

Syntax



OL805423

Description

The **tar** command writes files to or retrieves files from an archival storage medium. The **tar** command looks for archives on the default device (usually tape), unless you specify another device with the **-f** flag. File names must not be longer than 100 bytes and must not contain blanks. Characters following the first blank are ignored.

When writing to an archive, **tar** uses a temporary file (**/tmp/tar***) and maintains in memory a table of files with several links. You will receive an error message if **tar** cannot create the temporary file, or if there is not enough memory available to hold the link tables.

Notes:

1. When the storage device is an ordinary file or a block special file, **-u** and **-r** flags backspace. However, raw magnetic tape devices do not support backspacing. So when the storage device is a raw magnetic tape, the **-u** and **-r** flags rewind the tape, open it, and then read it again.
2. Records are one block long on block magnetic tape, but they are typically less than half as dense on raw magnetic tape. As a result, although a blocked raw tape must be read twice, the total amount of tape motion is less than it is when reading one-block records from a block magnetic tape once.

3. The structure of a streaming tape device does not support the addition of information at the end of a tape. Consequently when the storage device is a streaming tape, the **-u** and **-r** flags are not valid options. An attempt to use these flags results in the error message `tar: Update and Replace options not valid for a streaming tape drive.`
4. There is no way to ask for any occurrence of a file other than the last.
5. There is no recovery from tape errors.

Flags

You must supply one of the following five function flags to control the actions of **tar**:

c Creates a new archive and writes the *file* at the beginning of the archive.

r Writes the *file* at the end of the archive. Since the structure of a streaming tape device does not support the addition of information at the end of a tape, this option is not a valid flag when the archival storage device is a streaming tape.

t Lists the files in the order in which they appear in the archive. Files may appear more than once.

u Adds *file* to the end of the archive only if it is not in the archive already or if it has been modified since it was written to the archive. Since the structure of a streaming tape device does not support the addition of information at the end of a tape, this is not a valid flag when the archival storage device is a streaming tape.

x Extracts *file* from the archive. If you specify a *directory*, **tar** extracts all files in that directory from the archive. If you do not specify a *file* or a *directory*, **tar** extracts all of the files from the archive. When an archive contains multiple copies of the same file, **tar** extracts only the last one and overwrites all earlier ones. If you have superuser authority (see "**su**" on page 1026), **tar** creates all files and directories with the same user and group IDs as on the tape. If you do not have superuser authority, the files and directories have your user and group IDs.

The other optional flags to **tar** are listed below. In all cases, a directory parameter refers to all the files and subdirectories, recursively, within that directory. Flags without corresponding parameters may appear separately or be grouped together. Flags that take parameters may have them adjacent to the flag letter or as the entire following argument.

-b*blocks* Specifies the number of 512-byte blocks per record. The default is 16, which is appropriate for tape records. Due to the size of inter-record gaps, tapes written with large blocking factors can hold much more data than tapes with only one block per record.

The block size is determined automatically when tapes are read (function flags **-x** or **-t**). When archives are updated with the **-u** and **-r** functions, the existing record size is used. The **tar** command writes archives using the specified *blocks* value only when creating new archives with the **-c** flag.

For output to ordinary files with the **-f** flag, you can save disk space by using a blocking factor that matches the size of disk blocks (for example, **-b4** for 2048-byte disk blocks). Ordinary files must be read using the same blocking factor used when they were created.

-C*blocks*
: Allows **tar** to use very large clusters of blocks when it deals with streaming tape archives. Note, however, that on input, **tar** cannot automatically determine the block size of tapes with very long block sizes created with this flag. In the absence of a **-C***num* argument, the largest block size that **tar** can automatically determine is 20 blocks.

-d
: Makes separate entries for directories, blocks and character special files, and FIFOs. Normally, **tar** writes only ordinary files to an archive, and extracts only ordinary files and the directories required to contain them as determined by the path names in the archive. When writing to an archive with the **-d** flag, **tar** makes it possible to preserve the directory permission codes and to restore empty directories, special files, and FIFOs with the **-x** flag.

Note: Although anyone can archive special files, only a user with superuser authority can extract them from an archive.

-f*file*[**-***num*]
: Uses *file* as the archive to be read or written. When this flag is not specified, **tar** uses a system-dependent default file name of the form **/dev/rmt***?*. If the *file* specified is **-** (minus), **tar** writes to standard output or reads from standard input. If you write to standard output, the **-c** flag must be used (see example).

If you specify *num*, **tar** provides automatic spill over from one archive storage unit to another. This feature allows the operator of a system with multiple tape drives to use multitape archives without having to change tapes. For example, **-f/dev/rmt0-2** writes or reads **/dev/rmt0**, followed by **/dev/rmt1**, and then **/dev/rmt2** before requesting that additional volumes be mounted.

-h
: Ignores header checksum errors. The **tar** command writes a file header containing a checksum for each file in the archive. When this flag is not specified, the system verifies the contents of the header blocks by recomputing the checksum, and aborts with a `directory checksum error` when a mismatch occurs. When this flag is specified, **tar** logs the error and then scans forward until it finds a valid header block. This permits restoring files from later volumes of a multivolume archive without reading earlier volumes.

-i*inputlist*
: Writes the files listed in the *inputlist* file to the archive. The *inputlist* should contain one file name per line. Files from *inputlist* are not treated recursively. If you include the name of a directory in *inputlist*, **tar** does not write that directory's subdirectories to the tape, only that directory's files. If you also list files or directories on the command line, the contents of *inputlist* are included after **tar** has written all the files or the directories and their subdirectories to the archive.

-l
: Writes error messages to standard output if **tar** cannot resolve all of the links to the files archived. When you do not specify this flag, the system does not display these messages.

-m
: Uses the time of extraction as the modification time. The default is to preserve the modification time of the files.

-s *blocks***b**
-s *feet*
-s *feet @density*
: Specifies the number of 512-byte *blocks* per volume (first format), independent of the tape blocking factor. You can also specify the size of the tape in feet by using the second form, and **tar** assumes a default *density*. The third form allows you to specify both tape length and density. Feet are assumed to be 11 inches long to be conservative. This flag lets you deal more easily with multivolume tape archives, where **tar** must be able to determine how many blocks fit on each volume.

  Note: Tape drives vary in density capabilities. The *density* parameter calculates the amount of data a system can fit on a tape. This allows the correct amount of data to be written to a tape when using tape drives other than the IBM 6157 tape drive, which has a density of 700.

-v
: Lists the name of each file as it is processed. With the **-t** flag, **-v** gives more information about the tape entries, including file sizes, times of last modification, UID, and GID, and permissions.

-w
: Displays the action to be taken followed by the file name, then waits for user confirmation. If the response begins with a y or Y, then the action is performed. If the response is not affirmative, then the file is ignored.

Japanese Language Support Information

An affirmative response in Japanese Language Support matches one of the elements in the environment variable **YESSTR**.

-num Uses **/dev/rmt***num* instead of the default. For example, **-2** is the same as **-f/dev/rmt2**. In AIX systems with multidensity tape drives, this flag allows selecting a particular density. The default unit is system dependent and is chosen to match the default density, as described under the **-s** flag.

Examples

1. To write **file1** and **file2** to a new archive on the default tape drive:

   ```
   tar -c file1 file2
   ```

2. To extract all files that are in the **/tmp** directory from the archive file on the tape device **/dev/rmt2** and use the time of extraction as the modification time:

   ```
   tar -xm -f/dev/rmt2 /tmp
   ```

3. To create a new archive file that contains **file1** and pass the archive file to the **dd** command to be written to the device **/dev/rmt1**:

   ```
   tar -cvf - file1 | dd of=/dev/rmt1
   ```

4. To display the names of the files in the disk archive file **out.tar** on the current directory:

   ```
   tar -vtf out.tar
   ```

5. To expand the compressed archive file **fil.tar.z**, pass the file to the **tar** command, and extract all files from the expanded archive file:

   ```
   pcat fil.tar.z | tar -xvf -
   ```

Files

/dev/rmt?
/tmp/tar*

Related Information

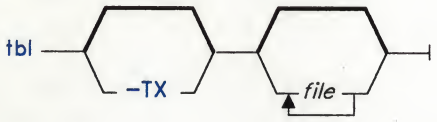
The discussion of Japanese Language Support in *Japanese Language Support User's Guide.*

tbl

Purpose

Formats tables for the **nroff, troff** and **troff** commands.

Syntax



OL805222

Description

The **tbl** command is a preprocessor that formats tables for **nroff** and **troff**. It reads the specified *file*s or, if you do not specify any file names or you specify a - (minus) as a file name, it reads standard input. The input is copied unchanged to standard output, except for text between lines containing **.TS** and **.TE**. This text describes tables, and is reformatted by **tbl**. The **.TS** and **.TE** lines are not altered by **tbl**. For more detailed information on how to format text for **tbl**, see *Text Formatting Guide*.

Note: When **tbl** is used with **eqn** or **neqn**, **tbl** should come first to minimize the volume of data passed through pipelines.

At the start of **tbl** text, you should include a line containing **.TS**. You can follow this with a line containing global options. The available global options are:

center	Centers the table (the default is left-adjusted).
expand	Makes the table as wide as the current line length.
box	Encloses the table in a box.
doublebox	Encloses the table in a double box.
allbox	Encloses each item of the table in a box.
tab (*x*)	Uses the character *x* instead of a tab to separate items in a line of input data.
linesize (*n*)	Sets lines and rules for boxes in point size *n*.
delim (*x*,*y*)	Sets *x* and *y* as the **eqn** and **neqn** text delimiters.

End the list of global options with a ; (semicolon).

After the global options, enter lines describing the format of each row in the table. Each format line (except the last) describes one row of the table. The last one describes all remaining rows of the table. This must end with a period to indicate that it is the end of the format specification. Each column of the table is described by a single keyletter.

The available keyletters are:

c Centers the item in the column.
r Right-adjusts the item in the column.
l Left-adjusts the item in the column.
n Adjusts the numerical items in the column to line up at the decimal point or right-adjusts them if there are no decimal points.
s Allows the previous item on the left to spill over into this column if the item is too wide for its column.
a Centers the longest line in this column and then left-adjusts all other lines in it with respect to the centered line.
^ Allows the item above to spill over into this column if the item is too large.
_ Replaces this entry with a horizontal line.
= Replaces this entry with a double horizontal line.

After the keyletter, you can enter specifiers that determine where vertical lines appear between columns, column width, inter-column spacing, and the font and point size of the item. The following table lists the legal specifiers.

Specifier	Meaning	Specifier	Meaning
e or **E**	Equal width columns	**w** or **W**	Minimum width column
f or **F**	Font change	**z** or **Z**	Zero width column
nnn	Column separation	*.xx*	Included **troff** request
p or **P**	Point size change	**|**	Vertical line
s or **S**	Spanned item	**||**	Double vertical line
t or **T**	Vertical spanning at top	**\^**	Vertical span
T{ . . . T}	Text block	**_**	Short horizontal line
u or **U**	Staggered columns	**\R***x*	Repeat character
v or **V**	Vertical spacing change		

Figure 11. tbl Column and Item Specifiers

The format lines are followed by lines containing data for the table. The last line consists of **.TE**. Within the data lines, data items are separated by tab characters, unless the global option, **delim** is used.

If a data line consists of only _ (underscore) or = (equal sign), a single or double line is drawn across the table at that point. If an entry in a data line consists of only _ or =, then that item is replaced by a single or double line.

Flag

-TX Uses only full vertical line motions, making the output suitable for line printers and other devices that do not have partial vertical line motions.

Related Information

The following commands: "**cw**, **checkcw**" on page 275, "**eqn**, **neqn**, **checkeq**" on page 395, "**mm**, **checkmm**" on page 663, "**mmt**, **checkmm**" on page 666, "**nroff, troff**" on page 709, and "**troff**" on page 710.

The **mm** and **mv** miscellaneous facilities in *AIX Operating System Technical Reference.*

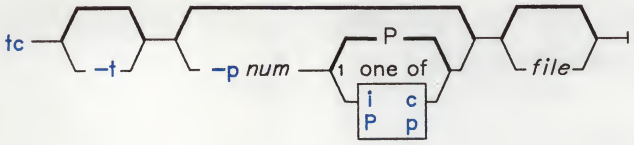
The discussion of **tbl** in *Text Formatting Guide.*

tc

Purpose

Simulates phototypesetter output for a Tektronix 4014 work station.

Syntax



OL805271

[1] Do not put a blank between these items.

OL805308

Description

The **tc** command interprets its input, either a *file* or standard input, as a **troff** document. It then simulates the typesetter output for a Tektronix 4014 work station with ASCII and APL character sets and writes the results to standard output (usually the work station display). The 16 typesetter sizes are mapped into the 4014's four sizes; the entire **troff** character set is drawn using the 4014's character generator, with overstruck combinations where necessary.

At the end of each page, **tc** waits for a new-line character from the keyboard before continuing to the next page. While it is waiting, the command e suppresses the screen erase before the next page. !*AIX-cmd* sends *AIX-cmd* to the shell.

There are no font distinctions in the display.

Flags

-p*num letter*	Sets page length to *num* and scale to *letter*. *letter* may include the scale factors **p** (points), **i** (inches), **c** (centimeters), and **P** (picas). The default is picas. Do not put a space between *num* and *letter*.
-t	Does not wait between pages (use in a pipeline).

Example

To use **tc** in a pipeline with **troff**:

```
troff -t chapter1 | tc
```

Related Information

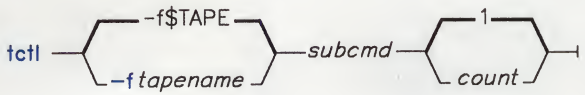
The following commands: "**sh**" on page 913, "**tplot**" on page 1079, "**troff**" on page 710, and "**4014**" on page 1264.

tctl

Purpose

Gives commands to streaming tape.

Syntax



OL805397

Description

The **tctl** command gives subcommands to a streaming tape device. If you do not specify the **-f** flag with *tapename*, the environment variable **TAPE** is used. If the environment variable does not exist, **tctl** uses the device **/dev/rmt4**. The *tapename* parameter must be a raw (not block) tape device. You can specify more than one operation with *count*.

Subcommands

eof **weof**	Writes *count* end-of-file markers at the current position on the tape.
fsf	Moves the tape forward *count* files.
fsr	Moves the tape forward *count* records.
rewind	Rewinds the tape. The *count* parameter is ignored. **Note:** It is sometimes necessary to issue a **reset** before issuing a **rewind** subcommand.
offline **rewoffl**	
reset	Places the tape drive off-line. The *count* parameter is ignored.
erase	Erases all contents on the tape and rewinds it.
retension	Moves the tape to the beginning, the end, and back to the beginning of the tape. If you have excessive read errors during a restore operation, you should run the **retension** subcommand. If the tape has been exposed to environmental extremes, you should run the **retension** subcommand before the save operation.

ras1 Performs a checksum on the tape drive.

ras2 Checks the capstan speed, verifies the operations of the BOT, EOT, and SAFE sensors, and writes a worst case pattern on the tape and attempts to verify the pattern.

Files

/dev/rmt?? The raw streaming tape interface.

Related Information

The following command: "**dd**" on page 301.

The **ioctl** system call and the **tape** and **environ** files in *AIX Operating System Technical Reference.*

tee

Purpose

Displays the output of a program and copies it into a file.

Syntax

OL805272

Description

The **tee** command reads standard input and writes the output of a program to standard output and copies it into *file* at the same time.

Flags

-a Adds the output to the end of *file* instead of writing over it.

-i Ignores interrupts.

Note: If you specify both flags, each must appear separately on the command line, preceded by a - (minus).

Examples

1. To view and save the output from a command at the same time:

   ```
   lint  program.c  |  tee  program.lint
   ```

   This displays the standard output of the command `lint program.c` at the work station, and at the same time saves a copy of it in the file `program.lint`. If `program.lint` already exists, it is deleted and replaced.

2. To display and append to a file:

```
lint  program.c  |  tee  -a  program.lint
```

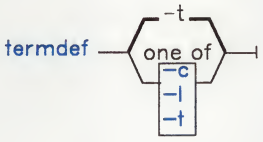
This displays the standard output of `lint program.c` at the work station and at the same time appends a copy of it to the end of `program.lint`. If the file `program.lint` does not exist, it is created.

termdef

Purpose

Queries terminal characteristics.

Syntax



OL805454

Description

The **termdef** command identifies the current display type, the active lines setting, or the current columns setting, thus simplifying the task of resetting the lines and columns when you switch fonts or of resetting the **$TERM** environment variable when you switch displays. The **terminfo** file defines the default number of lines and columns for each display, but the lines and columns can change depending upon which font is currently active. In addition, the **$TERM** environment variable does not automatically reflect the display currently being used. If you are using a display other than the **ibm5151**, you must explicitly reset this variable to access the **terminfo** correctly.

Flags

-c Returns the current column value.

-l Returns the current lines value.

-t Returns the name of the current display (this is the default action).

Example

To set environment variables according to the values of the currently active font and display, add the following lines to the **/etc/rc** file:

```
TERM=`termdef`
COLUMNS=`termdef  -c`
LINES=`termdef  -l`
export TERM LINES COLUMNS
```

Related Information

The following command: "**display**" on page 332.

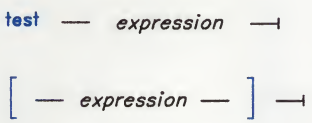
The **terminfo** file and the **hft** special file in *AIX Operating System Technical Reference.*

test

Purpose

Evaluates conditional expressions.

Syntax



OL805273

Description

The **test** command evaluates *expression* and, if its value is true, returns a zero (true) exit value. otherwise it returns a nonzero (false) exit value; **test** also returns a nonzero exit value if there are no parameters.

Note: In the second form of the command, that is the one that uses square brackets ([ ]), rather than the word **test**, the brackets must be surrounded by blanks.

Functions

All the functions and operators are separate parameters to **test**. The following functions are used to construct *expression*:

-r *file*	True if *file* exists and has read permission.
-w *file*	True if *file* exists and has write permission.
-x *file*	True if *file* exists and has execute permission.
-f *file*	True if *file* exists and is a regular file.
-d *file*	True if *file* exists and is a directory.
-c *file*	True if *file* exists and is a character special file.
-b *file*	True if *file* exists and is a block special file.
-p *file*	True if *file* exists and is a named pipe (FIFO).
-u *file*	True if *file* exists and its set-user-ID bit is set.

-g *file*	True if *file* exists and its set-group-ID bit is set.
-k *file*	True if *file* exists and its ***sticky bit*** is set.
-s *file*	True if *file* exists and has a size greater than zero.
-t [*filedescr*]	True if the open file with file descriptor number *filedescr* (1 by default) is associated with a work station device.
-z *s1*	True if the length of string *s1* is zero.
-n *s1*	True if the length of the string *s1* is nonzero.
s1 = *s2*	True if strings *s1* and *s2* are identical.
s1 != *s2*	True if strings *s1* and *s2* are not identical.
s1	True if *s1* is not the null string.
n1 **-eq** *n2*	True if the integers *n1* and *n2* are algebraically equal. Any of the comparisons **-ne, -gt, -ge, -lt,** and **-le** can be used in place of **-eq**.

These functions can be combined with the following operators:

!	Unary negation operator.
-a	Binary AND operator.
-o	Binary OR operator (**-a** has higher precedence than **-o**).
\(*expression* **\)**	Parentheses for grouping.

Examples

1. To test whether a file exists and is not empty:

```
if test ! -s "$1"
then
   echo $1 does not exist or is empty.
fi
```

   If the file specified by the first positional parameter to the shell procedure does not exist, this displays an error message. If $1 exists, it displays nothing. Note that there must be a space between **-s** and the file name.

   The double quotation marks around $1 ensure that the test will work properly even if the value of $1 is the empty string. If the double quotation marks are omitted and $1 is the empty string, **test** displays the error message `test: parameter expected`.

2. To do a complex comparison:

```
if [ $# -lt 2  -o  ! -s "$1" ]
then
   exit
fi
```

If the shell procedure was given fewer than two positional parameters or the file specified by $1 does not exist, then this exits the shell procedure. The special shell variable $# represents the number of positional parameters entered on the command line that started this shell procedure. For more details, see "Shell Variables and Command-Line Substitutions" on page 917.

Related Information

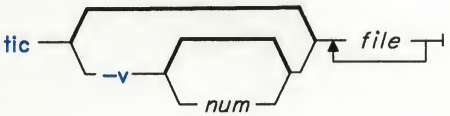
The following commands: "**find**" on page 422 and "**sh**" on page 913.

tic

Purpose

Translates **terminfo** files from source to compiled format.

Syntax



OL805340

Description

The **tic** command translates **terminfo** files from the source format into the compiled format. **tic** places the results in the directory **/usr/lib/terminfo**. If the environment variable **TERMINFO** is set, the results are placed there instead of in **/usr/lib/terminfo**.

The **tic** command compiles all terminfo descriptions in *files*. When **tic** finds a **use=** field, it searches first the current file, then the master file, **./terminfo.src**.

The total compiled entries cannot exceed 4096 bytes and the name field cannot exceed 128 bytes.

Flag

-v*num* Writes trace information on the progress of **tic**. *num* is an integer that increases the level of the verbosity.

Files

/usr/lib/terminfo/?/* Compiled terminal capability database.

Related Information

The **curses** subroutine and the **terminfo** file in *AIX Operating System Technical Reference*.

time

Purpose

Times the execution of a command.

Syntax

OL805274

Description

The **time** command times the execution of the named *command*. **time** writes to standard error the elapsed time of the command, the system time used, and the execution time, in seconds.

Examples

1. To measure the time required to run a program:

   ```
   time  a.out
   ```

   This runs the program **a.out** and writes to the standard error output the amount of real, system, and user time that it uses:

   ```
   real        10.5
   user         0.3
   sys          3.6
   ```

2. To save a record of the **time** information in a file:

   ```
   time  a.out  2> a.time
   ```

Related Information

The following commands: "**csh**" on page 225 and "**timex**" on page 1069.

Note: The **csh** command contains a built-in subcommand named **time**. The command and subcommand do not necessarily work the same way. For information on the subcommand, see the **csh** command.

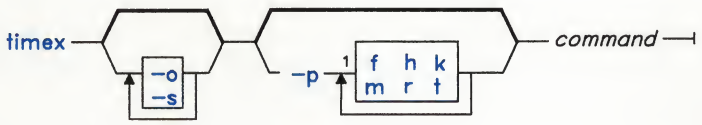
The **times** system call in *AIX Operating System Technical Reference*.

timex

Purpose

Times a command, and reports process data and system activity.

Syntax



[1]Do not put a blank between these items.

OL805275

Description

The **timex** command reports, in seconds, the elapsed time, user time, and system execution time for *command* where *command* is a local command. With flags specified, **timex** can list or summarize process accounting data for *command* and all of its children, and report total system activity during the execution interval. Output is written to standard error. The system uses the **/usr/adm/pacct** to select process records associated with *command* and includes background processes having the same user ID, work station ID, and execution time window.

Flags

-o Reports the total number of blocks read or written and total characters transferred by *command* and all its children.

-p Lists process accounting records for *command* and all its children. The number of blocks read or written and the number of characters transferred are always reported. The **f**, **h**, **k**, **m**, **r**, and **t** arguments, defined in the **acctcom** command, modify the other data items reported.

-s Reports total system activity that occurred during the execution of *command*. All the data items listed in the **sar** command are reported.

Related Information

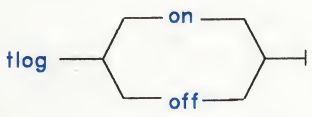
The following commands: "**acctcom**" on page 20 and "**sar**" on page 867.

tlog

Purpose

Stops or restarts sending of terminal I/O to a daemon.

Syntax



AJ2FL106

Description

The **tlog** command stops and restarts the sending of terminal I/O data to the message queue of the terminal-logging daemon, **tlogger**. **tlog off** stops I/O from going to the daemon. **tlog on** restarts the sending of the I/O.

The **tlog** command uses the **TCLOG ioctl ()** system call to stop and restart the flow of I/O to the daemon.

Files

/etc/tlogger Daemon writes terminal data to a log file.

Related Information

The following commands: "**tlogger**" on page 1072.

tlogger

Purpose

Gathers I/O from a terminal and writes it to a log file.

Syntax

AJ2FL105

Description

The terminal-logging daemon **tlogger** collects data read or written to its associated terminal and writes that data to a log file. Each time the daemon is started, the contents of the current log file replace the backup log file. A new current log file is created with permissions set to allow read and write by the owner.

The associated terminal is identified in the following manner: Standard error is assumed to be the correct terminal if it is a terminal device (**isatty()** returns true). Otherwise, the process's **usrinfo** is used to identify the login terminal, and that device is used.

The **tlogger** daemon creates a message queue, and passes that queue ID to the associated terminal using the **ioctl TCLOG** system call. The daemon then loops waiting on message queue data; it writes any message queue data it receives to the end of the current log file. The terminal log daemon will catch all signals (except **SIGKILL**). On receipt of a signal, the daemon issues an **ioctl TCLOG** to its associated terminal to turn off logging. This causes the terminal to stop sending log messages. The daemon then removes the message queue and exits. The daemon also terminates if it can no longer write to the log file due to file size constraints. In this case, an error message is written to standard error.

The **tlogger** daemon should be started in the background either from **/etc/rc**, or from the command line. This starts the terminal sending its I/O data to the daemon. The **tlog** command can then be used to stop or restart the sending of terminal I/O to the daemon. The daemon itself may be terminated with the **kill** command, but would ordinarily continue to run until shutdown occurs.

Notes:

1. **SIGKILL** should not be used to stop the daemon, since cleanup of system resources cannot be done in that case.
2. It may be necessary to prevent passwords from showing up in the terminal logs. You can prevent the system from logging passwords by having the **getpass()** subroutine turn off terminal logging while it is reading the password. The **login**, **adduser**, **newgrp**, and **passwd** commands use this subroutine.

Flags

-b *filename*	Specifies a file to be used as the backup log file. The default backup file is **/usr/adm/ras/tlogfile.bk**.
-c *filename*	Specifies a file to be used as the current log file. The default current file is **/usr/adm/ras/tlogfile**.

Files

/etc/rc	System startup file.
/usr/adm/ras/tlogfile	Default current log file.
/usr/adm/ras/tlogfile.bk	Default backup log file.

Related Information

The following commands: "**tlog**" on page 1071, "**shutdown**" on page 946, and "**kill**" on page 552.

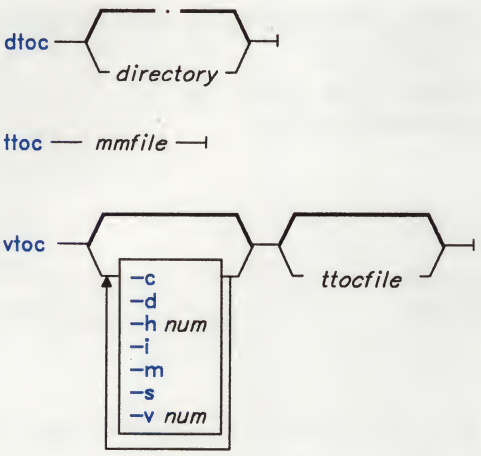
The **ioctl** system call and the **getpass** subroutine in *AIX Operating System Technical Reference.*

toc

Purpose

Provides graphical table of contents routines.

Syntax



OL777076

Description

All of the commands listed below reside in **/usr/bin/graf** (see "**graphics**" on page 497).

dtoc

The **dtoc** command makes a textual table of contents, **TTOC**, of all subdirectories beginning at *directory* (by default the current directory.). The list has one entry per directory. The entry fields from left to right are level number, directory name, and the number of ordinary readable files in the directory. **dtoc** is useful in making a visual display of all or parts of a file system. The following will make a visual display of all the readable directories under the root directory (/):

```
dtoc / ¦ vtoc ¦ td
```

ttoc

Output is the table of contents generated by the **.tc** macro of the **mm** command translated to **TTOC** format. The input is assumed to be a **mm** file that uses the .H family of macros for section headers. If no *file* is given, the standard input is assumed.

vtoc

The **vtoc** command produces a **GPS** describing a hierarchy chart from a **TTOC**. The output drawing consists of boxes containing text connected in a tree structure. If no *file* is given, the standard input is assumed. Each **TTOC** entry describes one box and has the form:

id[line-weight,line-style]"text"[mark]

where:

id is an alternating sequence of numbers and dots. The *id* specifies the position of the entry in the hierarchy. The *id* **0.** is the root of the tree.

line-weight is either:

n, normal-weight; or
m, medium-weight; or
b, bold-weight.

line-style is either:

so, solid-line;
do, dotted-line;
dd, dot-dash line;
da, dashed-line; or
ld, long-dashed

text is a character string surrounded by quotes. The characters between the quotes become the contents of the box. To include a quote within a box, it must be escaped (\").

mark is a character string (surrounded by quotes if it contains spaces). To include a dot within a box, it must be escaped (\.). The string is put above the top right corner of the box. To include either a quote or a dot within a *mark* it must be escaped.

Entry example:

```
1.1b,da"ABD" DEF
```

Entries may span more than one line by escaping the new-line (**\new-line**).

Comments are surrounded by the **/*,*/** pair. They can appear anywhere in a **TTOC**.

Flags

c Uses text as entered (default is all upper case).

d Connects the boxes with diagonal lines.

h*num* Sets horizontal interbox space to *num*% of box width.

i Suppresses the box *id*.

m Suppresses the box *mark*.

s Do not compact boxes horizontally.

v*num* Vertical interbox space is *num*% of box height.

Related Information

The following command: "**graphics**" on page 497.

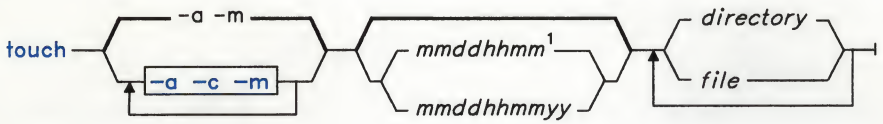
The **gps** file in *AIX Operating System Technical Reference*.

touch

Purpose

Updates the access and modification times of a file.

Syntax



[1]The current year is the default year.

OL805276

Description

The **touch** command updates the access and modification times of each *file* or *directory* named to the one specified on the command line. If you do not specify a time, **touch** uses the current time. If you specify a file that does not exist, **touch** creates a file with that name unless you request otherwise with the **-c** flag.

The environment variables **NLDATE** and **NLTIME**, if defined, specify the order of month and day in the date specification and of hour and minute in the time specification. Otherwise, these orders default to *mmdd* and *hhmm*.

The return code from **touch** is the number of files for which the times could not be successfully modified (including files that did not exist and were not created).

Flags

-a Changes only the access time.

-c Does not create the file if it does not already exist.

-m Changes only the modification time.

Examples

1. To update the access and modification times of a file:

   ```
   touch  program.c
   ```

   This sets the last access and last modification times of `program.c` to the current date and time. If `program.c` does not exist, **touch** creates an empty file with that name.

2. To avoid creating a new file:

   ```
   touch  -c  program.c
   ```

3. To update only the modification time:

   ```
   touch  -m  *.o
   ```

   This updates only the last modification times of the files in the current directory that end with `.o`. **touch** is often used in this way to alter the results of the **make** command.

4. To explicitly set the access and modification times:

   ```
   touch  -c  02171425  program.c
   ```

   This sets the access and modification dates to 14:25 (2:25 p.m.) February 17 of the current year.

Related Information

The following command: "**date**" on page 281.

The **utime** system call in *AIX Operating System Technical Reference.*

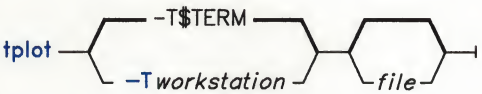
"Overview of International Character Support" in *Managing the AIX Operating System.*

tplot

Purpose

Produces plotting instructions for a particular work station.

Syntax



OL805277

Description

The **tplot** command reads plotting instructions from standard input or from *file*, if specified. (For more information about plotting instructions, see the **plot** file format *AIX Operating System Technical Reference*). **tplot** writes instructions suitable for the specified *workstation* to standard output. If *workstation* is not specified, the environment variable **TERM** is used. (For more information about environment variables, see the **environ** file in *AIX Operating System Technical Reference*).

Japanese Language Support Information

This command has not been modified to support Japanese characters.

Flag

-T*workstation* Uses the plotting instructions for *workstation*. The known *workstation* is:

lp IBM PC graphics printer

Files

/usr/lib/tcolor
/usr/lib/tprint

Related Information

The following commands: "**graph**" on page 494 and "**splp**" on page 975.

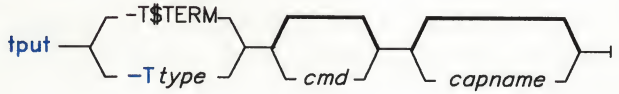
The **plot** subroutine and the **plot** file in *AIX Operating System Technical Reference.*

tput

Purpose

Queries the **terminfo** file.

Syntax



OL805398

Description

The **tput.** command uses the **terminfo** file to make terminal-dependent information available to the shell. The output of **tput.** is a string if the attribute *capname* (for capability name) is of type string or an integer if the attribute is of type integer. If the attribute is of type Boolean, **tput.** simply sets the exit value (0 for TRUE, 1 for FALSE), and produces no other output.

Flags

-T*type* Indicates the type of work station. Normally, the value of *type* is supplied by the environment variable **$TERM**.

capname Indicates the attribute from the **terminfo** file. For more information, see the **terminfo** file in *AIX Operating System Technical Reference.*

Examples

1. To echo the clear-screen sequence for the current work station:

   ```
   tput  clear
   ```

2. To display the number of columns for the current work station:

   ```
   tput  cols
   ```

3. To display the number of columns for the 450 work station:

   ```
   tput  -T450  cols
   ```

4. To set the shell variable `bold` to the highlight mode sequence for the current work station:

   ```
   bold=`tput smso`
   ```

   This might be followed by a prompt:

   ```
   echo "${bold}Please type in your name: \c"
   ```

5. To set the exit value to indicate if the current work station is a hardcopy terminal:

   ```
   tput hc
   ```

Files

/usr/lib/terminfo/?/*	Terminal descriptor files.
/usr/include/term.h	Definition files.
/usr/include/curses.h	

Related Information

The following command: "**stty**" on page 1018.

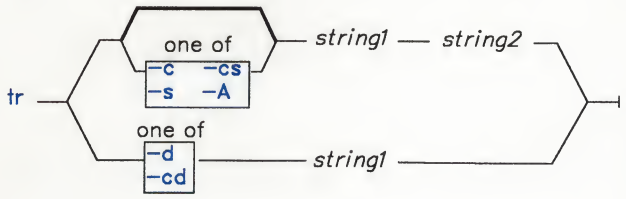
The **terminfo** file in *AIX Operating System Technical Reference.*

tr

Purpose

Translates characters.

Syntax



OL805278

Description

The **tr** command copies characters from the standard input to the standard output with substitution or deletion of selected characters. Input characters from *string1* are replaced with the corresponding characters in *string2*. **tr** cannot handle an ASCII NUL (\000) in *string1* or *string2*; it always deletes NUL from the input.

Abbreviations that can be used to introduce ranges of characters or repeated characters are:

[a-z] Stands for a string of characters whose ASCII codes run from character **a** to character **z**, inclusive.

[a*num] Stands for *num* repetitions of **a**. *num* is considered to be in decimal unless the first digit of *num* is **0**; then it is considered to be in octal.

Use the escape character \ (backslash) to remove special meaning from any character in a string. Use the \ followed by 1, 2, or 3 octal digits for the ASCII code of a character.

Japanese Language Support Information

You can use two octal sequences to specify a 2-byte kanji character. If you specify ranges of kanji characters, they are interpreted for translation as a string of kanji characters in ascending sequence in their binary representation.

Flags

-A Translates on a byte-by-byte basis. When you specify this flag, **tr** does not support extended characters.

-c Complements (inverts) the set of characters in *string1* with respect to the universe of characters whose ASCII codes are 001 through 377 octal, if you specify **-A**, and all characters, if you do not specify **-A**.

-d Deletes all input characters in *string1*.

-s Changes characters that are repeated output characters in *string2* into single characters.

Examples

1. To translate braces into parentheses:

```
tr  '{}'  '()'  <textfile  >newfile
```

   This translates each { to (and each } to). All other characters remain unchanged.

2. To translate lowercase characters to uppercase:

```
tr  '[a-z]'  '[A-Z]'  <textfile  >newfile
```

3. This is what happens if the strings are not the same length:

```
tr  '[0-9]'  '#'  <textfile  >newfile
```

   This translates each 0 to a # (number sign).

   Note: If the two character strings are not the same length, then the extra characters in the longer one are ignored.

4. To translate each digit to a #:

```
tr  '[0-9]'  '[#*]'  <textfile  >newfile
```

   The * tells **tr** to repeat the # enough times to make the second string as long as the first one.

5. To translate each string of digits to a single *num*:

```
tr  -s  '[0-9]'  '[#*]'  <textfile  >newfile
```

6. To translate all ASCII characters that are *not* specified:

```
tr  -c  '[  -~]'  '[A-_]?'   <textfile  >newfile
```

This translates each nonprinting ASCII character to the corresponding control key letter (\001 translates to A, \002 to B, etc.). ASCII DEL (\177), the character that follows ~ (tilde), translates to ?.

7. To create a list of the words in a file:

```
tr  -cs  '[a-z][A-Z]'  '[\012*]'   <textfile  >newfile
```

This translates each string of nonalphabetic characters to a single new-line character. The result is a list of all the words in `textfile`, one word per line.

Related Information

The following commands: "**ed**" on page 371 and "**sh**" on page 913.

The **ascii** file in *AIX Operating System Technical Reference.*

"Overview of International Character Support" in *Managing the AIX Operating System.*

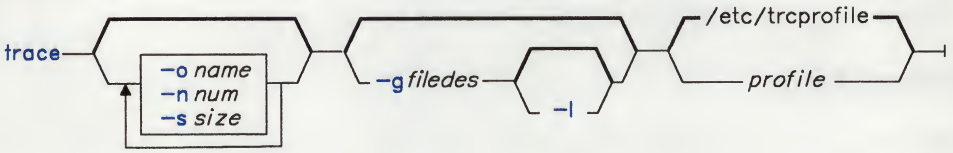
The discussion of Japanese Language Support in *Japanese Language Support User's Guide.*

trace

Purpose

Starts the trace function.

Syntax



OL805279

Description

The **trace** command starts the trace function in the background. This trace function provides a base for debugging the system. **trace** monitors the occurrence of selected events in the system and records on disk important data specific to each of these events. You can format this output with the **trcrpt** command.

Any user or program that needs the trace process enabled for debugging or error determination can start **trace**. When starting **trace**, you must provide a *profile*. This allows you to tailor the output of the trace session to individual needs. The default *profile* is **/etc/trcprofile**.

There may be more than one trace profile in the file system at a time. The trace profile contains the classes of events that you can select to trace, listed by event class and by a descriptive label. See "Example" on page 1089 for a sample profile. You may keep different profiles to trace different combinations of event classes. **trace** also takes additional information about the trace session from the configuration file **/etc/rasconf** (see *AIX Operating System Technical Reference* for a discussion of this file). You set the name and size of the output file in this configuration file.

In a multiuser environment, **trace** records all system events, not just events at one virtual terminal.

Japanese Language Support Information

This command has not been modified to support Japanese characters.

Flags

-g *filedes* Indicates that this is a ***generic*** trace session. Generic tracing applies only to the VRM. In this type of session, events to be recorded do not necessarily have a fixed event class, but are allocated to a temporary event channel by the trace device driver, **/de v/vrmtrace**. Thus, starting a generic trace does not require a trace profile. Generic traces are started and stopped by other processes, such as communications session managers. Therefore, the interface to the daemon is somewhat different. The **-g** flag is useful only when **trace** is started by another process.

The *filedes* parameter is a file descriptor from the parent process. **trace** writes this following information to this file descriptor:

- The process ID of the **trace** demon
- The address of the trace buffer
- The size (in bytes) of the trace buffer
- The temporary channel bit allocated to this event.

When tracing a generic event, the **trace** demon does not record its process ID so that it can be stopped by the **trcstop** command. Thus, more than one **trace** demon may be running at any time, but there may be as many as seven traces in the system at once (one normal trace and from one to six generic traces).

Use the **trc_start** and **trc_stop** subroutines to start and stop a generic trace.

-l Indicates that the VRM trace device driver should log only the last buffer filled before the **trace** demon stops. This flag is valid only during a generic trace (**-g**).

-o *name* Specifies the name of the log file into which the **trace** demon stores the trace data. For generic traces (**-g**), this name must be different from the default file name specified in the configuration file **/etc/rasconf**.

-n *num* Specifies the number of entries in the trace buffer. **trace** multiplies this number by the size of the entries (see the **-s** flag) and uses the resulting value to size the trace buffer. If you do not specify this flag, **trace** uses the buffer size specified in the configuration file **/etc/rasconf**.

-s *size* Specifies the size (in bytes) of the entries that the **trace** demon will be handling. The default size is 40 bytes. The size can be no less than 20, which is the number of bytes in the trace header for each entry. All entries must be the same size in a particular trace log file.

Example

```
*******************************************************************************
* SYSTEM TRACE PROFILE
*******************************************************************************
* To set trace on for an event class, remove the comment mark (*) from the
* first column of the line containing the event you wish to trace.
* Add a comment mark (*) in the first column of lines containing event types
* you wish to stop tracing.

***** Event
*     Type    Description

*****         Applications


*****         Kernel Extensions
*       36      Config

*****         Kernel System Calls
*       60      Shared Memory
*       61      Messages
*       62      Semaphores
*       63      Signals
*       64      Time
*       65      File System
*       66      File Handling
*       67      Directory Handling
*       68      Process

*****       VRM Components
*      100      SVC Handler
*      110      Async/5080 Peripherals
*      112      Async/5080 Peripheral Interrupts
*      113      Virtual Terminal Manager
*      114      Keyboard Interrupts
*      115      Locator Interrupts

*      150      User-Defined Events
```

Files

/etc/trcprofile	Default profile.
/usr/adm/ras/trcfile	Output file defined in **/etc/rasconf**.
/etc/rasconf	Configuration file.

Related Information

The following commands: "**trcstop**" on page 1093 and "**trcrpt**" on page 1091.

The **rasconf** configuration file in *AIX Operating System Technical Reference*.

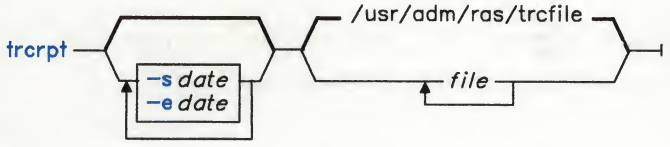
The discussion of **trace** in *AIX Operating System Programming Tools and Interfaces*.

trcrpt

Purpose

Formats a report from the trace log file.

Syntax



OL805280

Description

The **trcrpt** command writes to standard output a chronological listing in readable format of the trace log *file* or *file*s specified. You can specify a maximum of ten log files. If you do not specify any files, **trcrpt** reads **/etc/rasconf** for a file name. This name is usually **/usr/adm/ras/trcfile**.

Japanese Language Support Information

This command has not been modified to support Japanese characters.

Flags

-e *date* Ends the report time with entries on or before *date*. The format of *date* is the same as the **date** command, *MMddhhmmyy*.

-s *date* Starts the report with entries on or later than *date*. The format of *date* is the same as the **date** command, *MMddhhmmyy*. If you do not specify this flag, **trcrpt** formats the entire log file.

Example

To format a trace log file:

```
trcrpt -s0109100384 -e0109100584 /u/dave/trc_log  | print
```

This formats the log file /u/dave/trc_log, starting with entries from January 09, 1984 at 10:03 and ending at 10:05. It pipes the formatted output to the print queue.

Files

/usr/adm/ras/trcfile	Default log file.
/etc/trcfmt	Trace format file.
/usr/adm/ras/.trcevents	Trace event types table.

Related Information

The following commands: "**trace**" on page 1086 and "**trcstop**" on page 1093.

The **rasconf** file in *AIX Operating System Technical Reference.*

The discussion of **trcrpt** in *AIX Operating System Programming Tools and Interfaces.*

trcstop

Purpose

Stops the trace function.

Syntax

OL805223

Description

The **trcstop** command sends a Software Terminate signal to the **trace** background process. This gracefully ends **trace** and forces cleanup.

Japanese Language Support Information

This command has not been modified to support Japanese characters.

Files

/tmp/trc_PIDs

Related Information

The following commands: "**trace**" on page 1086 and "**trcrpt**" on page 1091.

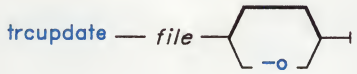
The discussion of **trcstop** in *AIX Operating System Programming Tools and Interfaces.*

trcupdate

Purpose

Updates trace format templates.

Syntax



OL805399

Description

The **trcupdate** command adds, replaces, or deletes trace report format templates in the files **/etc/trcfmt** and **/usr/adm/ras/.trcevents** and event types in the file **/etc/trcprofile**. **trcupdate** creates three undo files in the current directory named *file***.undo.trc**, **.trcevents.undo.evt**, and *file***.undo.pro**. These undo files can be used as input to **trcupdate** with the **-o** (override) flag to undo the changes **trcupdate** has just made.

The **trcupdate** command reads three files named *file***.trc**, *file***.evt**, and *file***.pro**. The **trc** file contains trace format templates; the **evt** file contains trace event types and their corresponding hook IDs; the **pro** file contains the event type line for the trace profile.

The first field of each template contains an operator:

+ To add or replace a template

- To delete a template.

If the operation is +, then the following fields contain the template to be replaced. The hook ID of the template is also added to the **/usr/adm/ras/.trcevents** file, and the event type line is added to the trace profile **/etc/trcprofile**. If the operation is a -, then the second field contains the hook ID of the template to delete. That hook ID is also deleted in **/usr/adm/ras/.trcevents**, and the event type line is deleted from **/etc/trcprofile**.

When adding or replacing, **trcupdate** compares the version numbers of each input template with the version number of the existing template of the same hook IDs. If the version number of the input template is later, it replaces the old template with the input template. If the template does not already exist, then it is added to the file. The input file *must* contain the identifier `* /etc/trcfmt` on the first line.

The *file*.**evt** file contains a table of trace system event types and hook IDs that fall under these types. **trcupdate** reads in the file **/usr/adm/ras/.trcevents** and adds in any hook IDs from *file*.**evt** that are not already accounted for or reassigns/deletes hook IDs to the event type given in the update file. The first line of the event/hook update file *must* be:

```
* /ras/.trcevents
```

or **trcupdate** rejects the input file.

The *file*.**pro** contains the lines that are to be added to or deleted from **/etc/trcprofile**. **trcupdate** reads **/etc/trcprofile** and adds or deletes the specified event type line from *file*.**pro**. The first line of the event type file *must* be: `* /etc/trcprofile` or **trcupdate** rejects the input file.

Japanese Language Support Information

This command has not been modified to support Japanese characters.

Flag

-o Does no version number checking.

Examples

1. The following is a sample **trc** file:

```
* /etc/trcfmt
+ 355 1.0 new_fmt
- 351
- 352
```

2. The following is a sample **evt** file:

```
* ras/.trcevents
350 355 356 357
```

Files

/etc/trcfmt
/usr/adm/ras/.trcevents
file.evt
file.undo.evt
file.trc
file.undo.trc
file.pro
file.undo.pro

Related Information

The following command: "**trcrpt**" on page 1091.

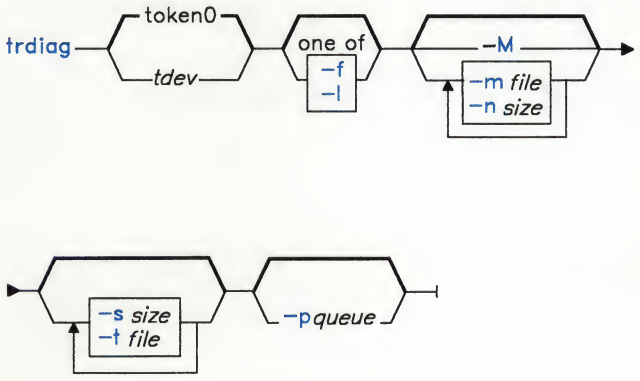
AIX Operating System Programming Tools and Interfaces.

trdiag

Purpose

Starts diagnostics on the Token-Ring Network.

Syntax



AJ2FL129

Description

The **trdiag** command starts the Token-Ring Diagnostics. Specify the device name of the token-ring adapter with *tdev* if you are not using **token0**.

Flags

-f Selects full error reporting. This flag cannot be used with the **-l** flag.

-l Selects limited error reporting. This flag cannot be used with the **-f** flag.

-M Requests logging of MAC frames. This flag uses the MAC frame log file and size specified in the `MAClog` stanza of the **/etc/trdconf** file.

-m *file* Specifies the MAC frame log *file*. The default file is specified in the `MAClog` stanza of the **/etc/trdconf** file.

-n *size* Specifies the *size*, in 1024 byte blocks, for the MAC frame log file. The default size is specified in the `MAClog` stanza of the **/etc/trdconf** file.

-p *queue* Specifies the printer *queue*. If you do not specify the **-p** flag the default printer queue is used.

-s *size* Specifies the *size*, in 1024 byte blocks, for the Token-Ring Diagnostic log file. The default is specified in the `trdlog` stanza of the **/etc/trdconf** file.

-t *file* Specifies the *file* for the Token-Ring Diagnostic log file. The default file is specified in the `trdlog` stanza of the **/etc/trdconf** file.

Examples

1. To start Token-Ring Diagnostics with one token-ring adapter installed:
   ```
   trdiag
   ```
2. To start Token-Ring Diagnostics with full error reporting enabled:
   ```
   trdiag -f
   ```

Files

/etc/trdconf Default configuration file.

Related Information

"Token-Ring Diagnostics" in *Managing the AIX Operating System*.

true

Purpose

Returns an exit value of zero.

Syntax

OL805064

Description

The **true** command returns a zero exit value. The **false** command returns a nonzero value. These commands are usually used in input to the **sh** command.

Example

To construct an infinite loop in a shell procedure:

```
while true
do
   date
   sleep 60
done
```

This shell procedure displays the date and time once a minute. To stop it, press **INTERRUPT (Alt-Pause)**.

Related Information

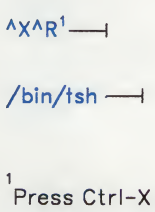
The following command: "**sh**" on page 913.

tsh

Purpose

Interprets commands in a trusted shell.

Syntax



AJ2FL136

Description

The AIX trusted shell (**tsh**) is a command interpreter which provides a subset of the functions of the **sh** command. The secure attention key (**Ctrl-X**, **Ctrl-R**) sequence (SAK) invokes the trusted shell. The trusted shell should be the login shell of the superuser. This command can also be issued by a program.

The following features are added to those of **sh** for the trusted shell (**tsh**):

- The **shell** command allows the user to return to the normal execution environment from **tsh**.
- The **logout** command allows the user to log off the system from **tsh**. This destroys any virtual terminals that the user may have opened.
- If **/bin/tsh** has the **tcb** attribute set, the user can only run trusted programs. A program must have the **tcb** attribute set to be executable by **tsh**.

The following **sh** features are not supported by **tsh**:

- **PATH** and **IFS** variable redefinition.
- Function/alias definition.

When started, this command interprets the **/etc/tsh** file. This file may contain a definition of the **PATH** variable.

Japanese Language Support Information

If Japanese Language Support is installed on your system, this command is not available.

Files

/etc/tsh_profile Contains initialization commands.

Related Information

The following commands: "**init**" on page 521 and "**shell**" on page 938

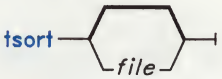
The discussion of the trusted path and **sak** in *Managing the AIX Operating System.*

tsort

Purpose

Sorts an unordered list of ordered pairs (a topological sort).

Syntax



OL805224

Description

The **tsort** command reads from *file* or standard input an unordered list of ordered pairs, it builds a completely ordered list, and writes it to standard output.

The input *file* should contain pairs of nonempty strings separated by blanks. Pairs of different items indicate a relative order. Pairs of identical items indicate presence, but no relative order. You can use **tsort** to sort the output of the **lorder** command.

If *file* contains an odd number of fields, **tsort** writes the error message `Odd data`.

Example

To create a subroutine library:

```
lorder  charin.o  scanfld.o  scan.o  scanln.o  \
|  tsort  |  xargs  ar  qv  libsubs.a
```

This creates a subroutine library named `libsubs.a` that contains `charin.o`, `scanfld.o`, `scan.o`, and `scanln.o`. The ordering of the object modules in the library is important. The **ld** command requires each module to precede all the other modules that it calls or references. The **lorder** and **tsort** commands together add the subroutines to the library in the proper order.

Suppose that `scan.o` calls `scanfld.o` and `scanln.o`. `scanfld.o` also calls `charin.o`. First, the **lorder** command creates a list of pairs that shows these dependencies:

```
charin.o charin.o
scanfld.o scanfld.o
scan.o scan.o
scanln.o scanln.o
scanfld.o charin.o
scanln.o charin.o
scan.o scanfld.o
```

Next, the | (vertical bar) sends this list to the **tsort** command, which converts it into the ordering needed:

```
scan.o
scanfld.o
scanln.o
charin.o
```

Note that each module precedes the module it calls. `charin.o`, which does not call another module, is last.

The second | then sends this list to **xargs**, which constructs and runs the following **ar** command:

```
ar qv libsubs.a scan.o scanfld.o scanln.o charin.o
```

This **ar** command creates the properly ordered library.

Related Information

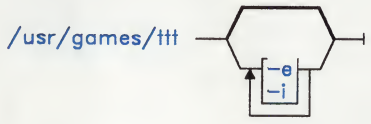
The following commands: "**ar**" on page 55, "**lorder**" on page 591, and "**xargs**" on page 1232.

ttt

Purpose

Plays tic-tac-toe.

Syntax



OL805282

Description

The **ttt** game plays the popular X and O game. This is a learning version, but it learns slowly. It loses nearly 80 games before completely mastering the game.

Flags

-e Increases the speed of the learning.

-i Displays the instructions prior to the start of the game.

To quit the game, press **INTERRUPT (Alt-Pause)** or **END OF FILE (Ctrl-D)**.

Files

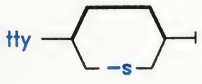
/usr/games/ttt.a Learning file.

tty

Purpose

Writes to standard output the full path name of your work station.

Syntax



OL805283

Description

The **tty** command writes the name of your work station to standard output.

Flag

-s Suppresses reporting the path name. The exit value has the following possible meanings:

0 Standard input is a work station.
1 Standard input is not a work station.
2 Invalid flags specified.

If your standard input is not a work station and you do not specify the **-s** flag, you get the message `not a tty`.

Examples

1. To display full path name of your work station:

```
tty
```

2. To test whether or not the standard input is a work station:

```
if tty -s
then
   echo 'Enter the text to print:' >/dev/tty
fi
print
```

If the standard input is a work station, this displays the message `Enter the text to print:` as a prompt and prints the text that the user types. If the standard input is not a work station, this displays nothing. It merely prints the text read from the standard input.

The `echo . . . >/dev/tty` displays the prompt on the screen even if you redirect the standard output of the shell procedure. This way the prompt is never written into an output file. The special file **/dev/tty** always refers your work station, although it also has another name like **/dev/console** or **/dev/tty2**.

turnon

Purpose

Turns on execute permission for games.

Syntax

OL805405

OL805406

Description

The **turnon** and **turnoff** commands are shell procedures that set the permission codes of files in the **/usr/games** directory. You must be operating with superuser authority to run this command.

The **turnon** command looks for files with permissions set to 000 and sets them to 111 (execute permission for all users).

The **turnoff** command looks for files in **/usr/games** whose permissions are set to 111 and sets these permissions to 000.

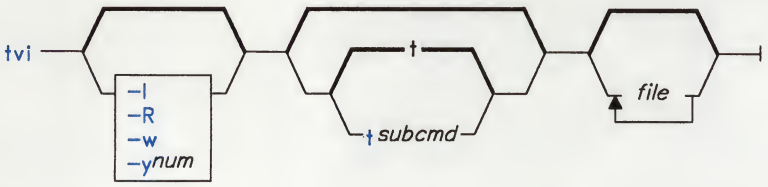
If you install any new games in the **/usr/games** directory, set their permissions to 111.

tvi

Purpose

Acts as trusted editor for system administration.

Syntax



OL805557

Description

The **tvi** command provides an editor that works with a subset of the functions of the **vi** editor. It creates files only when called by the user. When auditing is enabled, and a file is edited using this command, an audit record of the type **tvi** is created.

The following **vi** features are not supported by **tvi**:

- Shell escapes
- User-defined macros
- Key mapping
- Keeping customized changes
- The **-r** [*file*] and **-t** *tag* flags
- Preserve or recover text operations
- Tags

Japanese Language Support Information

If Japanese Language Support is installed on your system, this command is not available.

Related Information

The "**vi, vedit, view**" on page 1187 commands.

ugtable

Purpose

Creates, displays, and changes the Distributed Services Network Users/Groups Table.

Syntax

OL805469

Description

The **ugtable** command lets you build, examine, or modify the Distributed Services Network Users/Groups Table. Only members of the system group or users operating with superuser authority can use **ugtable** to change the state of the Distributed Services Network Users/Groups Table (see "**su**" on page 1026). Other users can use **ugtable** to browse the Network Users/Groups Table.

Related Information

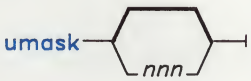
"Getting Started With Distributed Services Configuration Menus" in *Managing the AIX Operating System.*

umask

Purpose

Displays and sets file-creation permission code mask.

Syntax



OL805286

Description

The **umask** command sets your file-creation mask to *nnn*, three octal digits that represent the read/write/execute permissions for owner, group, and others, respectively. When you create a file, the system ANDs the complement of *nnn* to 777 for directories and 666 for files, in effect removing the corresponding permissions. (See "**chmod**" on page 160 for more information on file and directory permission codes.)

If you do not specify *nnn*, **umask** displays the current value of your file-creation permission code mask. The initial system mask (set in **/etc/profile**) is 022.

Examples

1. To display the current file creation mask:

   ```
   umask
   ```

2. To prevent other people from writing to your directories or files:

   ```
   umask 022
   ```

   This sets the file creation mask to 022, which takes away write permission for group members and others. Directories are created with the permission code 755. Files are created with 644.

3. To prevent other people from using your files:

   ```
   umask 077
   ```

   This sets the file creation mask to 077, which removes read, write, and execute permission for group members and others. Now files are created with permission code 600.

Related Information

The following commands: "**chmod**" on page 160, "**csh**" on page 225, and "**sh**" on page 913.

Note: The **csh** command contains a built-in subcommand named **ummask**. The command and subcommand do not necessarily work the same way. For information on the subcommand, see the **csh** command.

The **creat**, **chmod**, **mknod**, **open**, and **umask** calls in *AIX Operating System Technical Reference.*

The discussion of file permissions in *Using the AIX Operating System.*

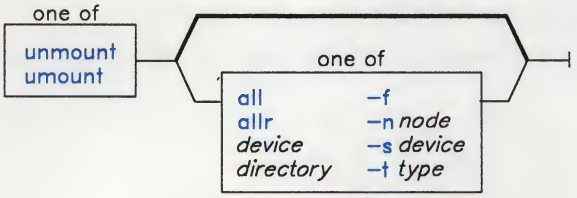
The discussion of tailoring the user environment in *Managing the AIX Operating System.*

umount, unmount

Purpose

Unmounts a previously mounted file system, directory, or file.

Syntax



OL805225

Description

The **umount** command unmounts a previously mounted file system, directory, or file. Processing on the file system, directory, or file completes and it is unmounted. Members of the system group and users operating with superuser authority can issue any **umount** command. Other users can unmount any directory or file if they have write permission to the mounted directory or file. For local mounts, you can specify the file system, directory, or file as either the *directory* or *device* on which it is mounted.

You can also specify one of the following parameters:

all Unmounts all mounted file systems.

allr Unmounts all remote mounted file systems.

Note: For remote mounts, specify the directory of the file as *directory*. If you specify **allr**, **umount** unmounts all remote mounts.

Flags

-f Forces the unmount of one or more virtual file systems. Use in a distributed services or NFS environment to free a client when the server is down and server path names cannot be resolved.

-n *node*	Specifies the node which holds the mounted directory that you want to unmount. For Distributed Services, *node* can be a nickname or a node ID. For NFS, *node* can be a host name or an alias. The **umount -n** *node* command unmounts all remote mounts made from *node*.
-s *device*	Prohibits the use of the **/etc/mnttab** file if it is damaged or not writable. If you use this flag, you must specify the name of the *device* to be unmounted.
-t *type*	Unmounts all stanzas in **/etc/filesystems** that contain **type** = *type* and are mounted. (*type* is a string value, such as `remote`.)

Note: You cannot use the **umount** command on a device that is in use. A device is in use if any file is open for any reason or if a user's current directory is on that device.

Examples

1. To unmount a diskette drive:

   ```
   umount /dev/fd0
   ```

2. To unmount the device mounted on **/diskette0**:

   ```
   umount /diskette0
   ```

3. To unmount all mounts from a remote node:

   ```
   umount -n nodeA
   ```

4. To unmount files and directories of a specific type:

   ```
   umount -t remote
   ```

   This unmounts all files or directories that have a stanza in the **/etc/filesystems** file that contains the attribute `type = remote`.

Files

/etc/filesystems	Descriptions of mountable file systems.
/etc/mnttab	Table of currently mounted file systems.

Related Information

The following command: "**mount**" on page 669.

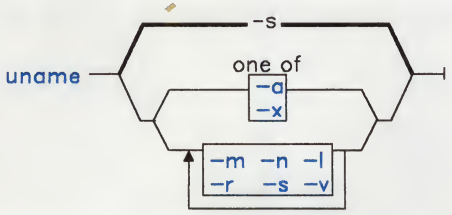
The **mount**, **umount**, **vmount**, **uvmount**, and **mntctl** system calls and the **mnttab** file in *AIX Operating System Technical Reference*.

uname

Purpose

Displays the name of the current operating system.

Syntax



OL805287

Description

The **uname** command writes to the standard output the name of the operating system that you are using.

Flags

-a Displays all information specified with the **-m**, **-n**, **-r**, **-s**, and **-v** flags.

-l Displays the LAN network number. (When Japanese Language Support is installed on your system, the **-l** flag is inactive.)

-m Displays the type of hardware running the system.

-n Displays the name of the node (this may be a name the system is known by to a uucp communications network).

-r Displays the release number of the operating system.

-s Displays the system name. (This flag is on by default.)

-S Sets the uucp node name to the arguments parameter list.

Note: You must be superuser to use this flag. No other flags are permitted.

-v Displays the operating system version.

-x Displays the information specified with the **-a** flag and the LAN network number. (When Japanese Language Support is installed, the **-x** flag is inactive.)

If you enter a flag that is not valid, **uname** exits with an error message, an error return status, and no output.

Example

To display the complete system name and version banner:

```
uname -a
```

To set the node name to `lance`:

```
uname -S lance
```

Related Information

The **uname** system call in *AIX Operating System Technical Reference.*

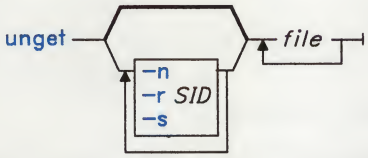
The discussion of Japanese Language Support in *Japanese Language Support User's Guide.*

unget

Purpose

Cancels a previous **get** command.

Syntax



OL805284

Description

The **unget** command allows you to restore a g-file created with **get -e** before the new delta is created, and therefore discarding the changes (see "**get**" on page 477 and "**delta**" on page 310). If you specify a - (hyphen) in place of *file*, standard input is read, and each line of standard input is interpreted as the name of an SCCS file. **unget** continues to take input until it reaches an end-of-file character, which is **Ctrl-D** if entered from the keyboard.

If you specify a directory in place of *file*, **unget** performs the requested actions on all SCCS files (those files with the **s.** prefix).

Flags

Each flag or group of flags applies independently to each named file.

-n Prevents the automatic deletion of the g-file. This flag allows you to retain the edited version of the file without making a delta.

-r*SID* Specifies the new delta that would have been created by the next use of the **delta** command. You must use this flag if you have two or more pending deltas to the file under the same login name. You can look at the p-file to see if you have more than one delta pending to a particular SID under the same login name. The *SID* specification must unambiguously specify only one SID to discard, or **unget** displays an error message and stops running.

-s Suppresses writing the deleted SID to standard output.

Example

To discard the changes you have made to an SCCS file after entering a **get -e**:

```
unget  s.prog.c
```

Related Information

The following commands: "**delta**" on page 310, "**get**" on page 477, and "**sact**" on page 862.

The **sccsfile** file in *AIX Operating System Technical Reference.*

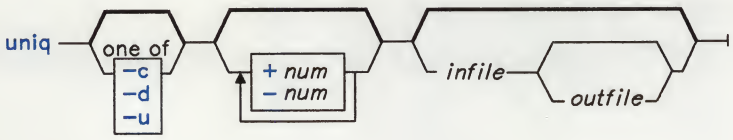
The discussion of SCCS in *AIX Operating System Programming Tools and Interfaces.*

uniq

Purpose

Deletes repeated lines in a file.

Syntax



OL805285

Description

The **uniq** command reads standard input or *infile*, compares adjacent lines, removes the second and succeeding occurrences of a line, and writes to standard output or the specified file *outfile*. *infile* and *outfile* should always be different files. Repeated lines must be on consecutive lines in order to be found. You can arrange them with the **sort** command (see page 958) before processing.

Flags

-c	Precedes each output line with a count of the number of times each line appears in the file. This flag supersedes **-d** and **-u**.
-d	Displays only the repeated lines.
-u	Displays only the unrepeated lines.
-num	Skips over the first *num* fields. A field is a string of nonspace, nontab characters separated by tabs and or spaces from adjacent data on the same line.
+num	Skips over the first *num* characters. Fields specified by *num* are skipped before characters.

Related Information

The following commands: "**comm**" on page 183 and "**sort**" on page 958.

units

Purpose

Converts units in one measure to equivalent units in another measure.

Syntax

OL805226

Description

The **units** command converts quantities expressed in one measurement to their equivalents in another. **units** is an interactive command. It prompts you for the unit you want to convert from and the unit you want to convert to (see "Examples" on page 1120). This command only does multiplicative scale changes. That is, it can convert from one value to another only when the conversion is done with a multiplication factor. For example, it cannot convert between degrees Fahrenheit and degrees Celsius, because 32 must be added or subtracted in the conversion.

You can specify a quantity as a multiplicative combination of units, optionally preceded by a numeric multiplier.

Indicate powers by suffixed positive integers and division by / (slash).

The **units** command recognizes **lb** as a unit of mass, but considers **pound** to be the British pound sterling. Compound names are run together (such as **lightyear**). Prefix British units differing from their American counterparts with **br** (**brgallon** for instance). The file **/usr/lib/unittab** contains a complete list of the units that the **units** command uses.

Most familiar units, abbreviations, and metric prefixes are recognized, together with the following:

pi	Ratio of circumference to diameter
c	Speed of light
e	Charge on an electron
g	Acceleration of gravity
force	Same as **g**
mole	Avogadro's number
water	Pressure head per unit height of water
au	Astronomical unit.

Japanese Language Support Information

This command has not been modified to support Japanese characters.

Examples

To start the **units** command, enter:

```
units
```

Now you can try the following examples. In these examples, the text that you enter is shown in **bold type** and the output from **units** is shown in non-bold type.

1. To display conversion factors:

```
you have: in
you want: cm
        * 2.540000e+00
        / 3.937008e-01
```

   The output from **units** tells you to multiply the number of inches by 2.540000e+00 to get centimeters, and to multiply the number of centimeters by 3.937008e-01 to get inches.

   These numbers are in standard exponential notation, so 3.937008e-01 means 3.937008×10^{-1}, which is the same as 0.3937008. The second number is always the reciprocal of the first. That is, $2.54 = 1 \div 0.3937008$.

2. To convert a measurement to different units:

```
you have: 5 years
you want: microsec
        * 1.577846e+14
        / 6.337753e-15
```

   The output shows that **5 years** equals 1.577846×10^{14} microseconds, and that one microsecond equals 6.337753×10^{-15} years.

3. To give fractions in measurements:

```
you have: 1|3 mi
you want: km
        * 5.364480e-01
        / 1.864114e+00
```

   The | (vertical bar) indicates division, so **1|3** means one-third. This shows that one-third mile is the same as 0.536448 kilometers.

4. To include exponents in measurements:

```
you have: 1.2-5 gal
you want: floz
        * 1.536000e-03
        / 6.510417e+02
```

The expression **1.2-5 gal** stands for 1.2×10^{-5}. Do *not* type an **e** before the exponent. This example shows that 1.2×10^{-5} (0.000012) gallons equal 1.536×10^{-3} (0.001536) fluid ounces.

5. To specify complex units:

```
you have: gram centimeter/second2
you want: kg-m/sec2
        * 1.000000e-05
        / 1.000000e+05
```

The units **gram centimeter/second2** mean "grams × centimeters ÷ second^2." Similarly, **kg-m/sec2** means "kilograms × meters ÷ sec^2," which is often read as "kilogram-meters per seconds squared." Note that you can show multiplication of units with a - (hyphen) or with a blank.

6. If the units you specify after "`you have`" and "`you want`" are incompatible:

```
you have: ft
you want: lb
conformability
        3.048000e-01 m
        4.535924e-01 kg
```

The message `conformability` means that the units you specified cannot be converted. Feet measure length, and pounds measure mass, so converting from one to the other doesn't make sense. Therefore, the **units** command displays the equivalent of each value in standard units.

In other words, this example shows that one foot equals 0.3048 meters and that one pound equals 0.4535924 kilograms. **units** shows the equivalents in meters and kilograms because the command considers these units to be "standard" measures of length and mass.

Files

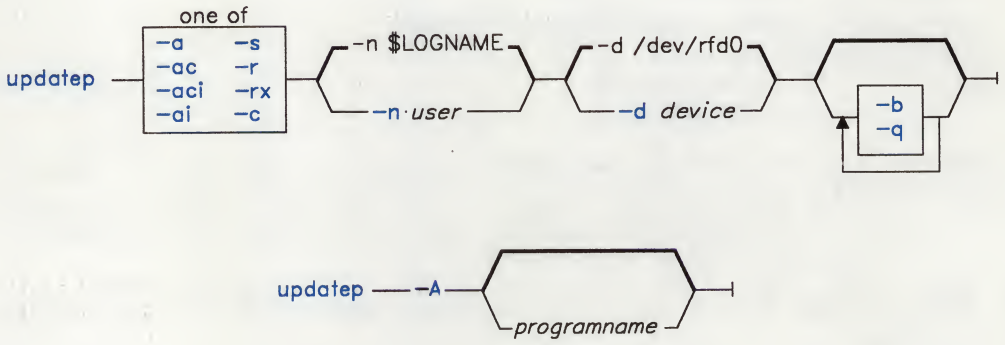
/usr/lib/unittab

updatep

Purpose

Updates one or more programs.

Syntax



OL805392

Description

Warning: Before you apply or reject an update, restart your system and make sure no other programs are running and no other work stations are enabled.

The **updatep** command controls the update process for one or more programs. It also lets you determine the status of pending program updates and provides documentation about the updates. In addition, **updatep** can provide a list of the committed updates for each licensed program and the information changed by the update. You must be a member of the system group or operating with superuser authority to run this command.

The **updatep** command supports an apply/commit/reject philosophy. To apply one or more programs, use the **-a** or the **-ai** flags. Then use either the **-c** flag to commit the program or the **-r** flag to reject the program. Normally you do not use **-r** until you have tested the program on your system. If you specify **-ac** or **-aci**, you can apply and commit in one operation. The **-r** flag must be used separately. During an apply, **updatep** normally saves the current versions of files that are being updated. If needed, these files can be used to do a recovery or reject.

You are responsible for reserving update save space in the **/usr** file system. The **updatep** command checks to insure there is adequate save space in **/usr** before it applies an update. If there is insufficient free space, **updatep** gives you the option of either ending the command or allowing it to continue. If you end the command, you can take action to increase the free space in your **/usr** file system. If you continue, no current versions of files are saved, and **updatep** automatically commits the update, even though you may not have requested a commit originally. Normally, you should reserve 4000 blocks (2 megabytes) of free space in the **/usr** file system for updates.

You cannot use **INTERRUPT (Alt-Pause)** to stop the **updatep** command. To stop **updatep**, press **QUIT WITH DUMP (Ctrl-V)**. Use only in extreme circumstances since the state of the system cannot be predicted. For example:

- The **write-verify** feature may be left on for all minidisks. See "**verify**" on page 1186.
- All terminals other than the console may be disabled. See "**pstart, penable, pshare, pdelay**" on page 791.
- Some update control files may need to be deleted.

Flags

-a Shows authorized program analysis reports (APARs) that have been fixed for the licensed program products (LPPs) specified if none are given. This flag then shows all installed LPPs.

-a[i] Applies the updates for one or more programs. If there is a pending update for any program on the system, **updatep** does not permit an apply. You must either commit or reject all pending updates before it accepts another update apply.

The **updatep** command asks you to select the program you wish to update. After you select a program, **updatep** runs the **inudocm** command for any specific update instructions. If it finds any, it copies them into the **/usr/lpp/***pgm-name***/ui.***vv.rr.llll.* file, where *vv* is the version, *rr* the release, and *llll* the level of the program. Review instructions before continuing. To restart the update procedure and ignore the check for existing update instructions, enter `updatep -ai` or `updatep -aci`.

The **updatep** command applies the update for each program by running **inuupdt** for each name. After each update, it deletes the **/usr/lpp/***pgm-name***/inst_updt** directory. It then runs **inudocm** to check for any update documentation. If there is information for a manual, **updatep** copies it into the **/usr/lpp/***pgm-name***/me.***vv.rr.lll* file and writes a message.

-A *programname*
Displays a record of the applied licensed programs on your system. When used with this flag **updatep** also displays the specific areas corrected in each

program. If you specify the name of the program, **updatep** displays information for only that program.

-b Runs the **bffcreate** command to create a backup format file from the distribution media. Then tells **updatep** to use the backup file as the distribution media for the update. Use this flag to update in a code service environment.

-c Commits a previous update apply. The **updatep** presents selection information for programs that have pending updates. You select the programs that you want to commit.

Any programs that you apply as a group must be committed as a group. Management control information about the update changes to indicate that the program is committed. **updatep** deletes the directory that contains the update recovery information, **/usr/lpp/***pgm-name***/inst_updt.save**.

-d *device* Specifies the input device name. The default input device is **/dev/rfd0**.

-n *user* Lets you specify a name in the program history file that is responsible for the program. The default is the value of the system variable **$LOGNAME**. If you specify *user*, the first eight nonblank characters are stored in the program history file.

-q Runs in quiet mode, suppressing most of the interactive queries.

-r Rejects a previous update apply for one or more programs. **updatep** presents selection information for the programs that have pending updates. You select the programs to reject.

Any programs that are grouped together by the system must be rejected or applied as a group. Specify **-r** without **-x** if you want automatic recovery of saved files. If you specify the **-x** flag, the management control information about the update reflects that the update is rejected, but **updatep** does not recover saved files. To recover the necessary files, look at the information in **/usr/lpp/***pgm-name***/inst_updt.save**. This flag should only be used by someone very knowledgeable about the system.

-s Writes status information about all pending program updates.

-x Cancels the automatic recovery of saved files. (Use with **-r**.)

The **updatep** command receives the following exit codes indirectly from **update** through **inuupdt**:

0 Normal return, no errors indicated.
2 Use the **sync** command to update the super blocks, i-nodes, and delayed block I/O and then restart the AIX Operating System.
3 Build the kernel, then update the superblocks, i-nodes, and delayed block I/O (sync) and shut down the AIX Operating System.

4 Use the **cfgaply** subroutine to build the kernel. Use the **sync** command to update the super blocks, i-nodes, and delayed block I/O and then restart the AIX Operating System.
5 Installation cancelled without errors.
6 Update superblocks, i-nodes, and delayed block I/O (sync), then shut down the AIX Operating System.
7 Update cancelled by update procedure; recovery needed.

If it receives any other exit code, it runs the recovery function **inurecv**. If the system cannot run **updatep**, it returns an exit code of 1.

Internal Commands

The **updatep** command uses the **inudocm** command for update documentation control. It uses the **inuupdt** command to apply an update to a single program. **inuupdt** runs a program-provided update procedure, **update**. **updatep** passes the following parameters to the **update** procedure:

- The full path name of the apply-list.
- The full path name of the device (file), where the update information is stored in **backup** format.

In addition to the commands discussed here, program-provided update procedures can use all of the internal commands discussed in "**installp**" on page 529. Since they are internal commands, they do minimum validation of input parameters. Their purpose is to provide common code for functions frequently needed by most program-provided procedures. Since these internal commands function as subcommands, they return exit values rather than issue error messages. However, messages may come from other system commands that they run. C Language programmers of update procedures that call these commands can use the **/usr/include/inu21.h** file to define the return codes for them.

inudocm

The **inudocm** command is normally used as an internal command to get copies of specific update instructions or manual errata information that you can print out. There can be cases, however, when you would enter this command from the command line (for example, if you misplace the manual errata information that came with a previous update). You must be a member of the system group or operating with superuser authority to run this command. When auditing is on, an audit record of the type, **inudocm** is created.

The **inudocm** command has the following syntax:

inudocm -eu [**-d** *device*] [*pgm-name*] [*level*] [**-f** *file*]

where *pgm-name* specifies the name of the program being checked. It must be specified unless you use the **-f** flag. It can be a maximum of eight characters. *level* specifies the current level of *pgm-name*. This value must be identical to the level value for the last

committed entry in **/usr/lpp/***pgm-name***/lpp.hist**. It must be specified unless you use the **-f** flag.

Flags

-d *device* Restores *file* from this *device*. The *device* variable must be the full path name of a device special file. The default *device* is **/dev/rfd0**. Do not specify this flag if you use the **-f** flag.

-e Requests the existing update documentation information for *pgm-name* from *level*. If you select this flag, **inudocm** uses the **ar x** command to extract the archive file **/usr/sys/inst_updt/***pgm-name***_erata**. If this file is not present, no information is available. **inudocm** extracts any level-dependent manual errata information files if there are any more recent than the current level. Selected files are moved to **/usr/lpp/***pgm-name***/me.***vv.rr.llll*.

-f *file* Identifies a file that already contains the *pgm-name* and the *level*. Only **updatep** itself should use this flag.

-u Requests the existing specific update instructions for *pgm-name* from *level*. If you select this flag, **inudocm** uses the **ar x** command to extract the archive file **/usr/sys/inst_updt/***pgm-name***_instr**. If this file is not present, no information is available. **inudocm** extracts any level-dependent specific update instruction files if there are any more recent than the current level. Selected files are moved to **/usr/lpp/***pgm-name***/ui.***vv.rr.llll*.

The **inudocm** command returns the following exit values:

0 Normal return, no error occurred.
1 The system cannot run **inudocm**.
2 Specific update instruction files were requested but not found.
4 Manual errata information was requested, but not found.
6 Specific update instructions and manual errata were both requested but not found.
201 An invalid flag was specified, or the first argument was not **-e**, **-eu**, or **-u**.
202 One or more parameters were missing.
204 Too many parameters were entered.
250 The *level* parameter did not contain exactly 4 characters, or they were not numeric.
251 An error occurred while attempting to restore **/usr/sys/inst_updt/control**.
253 The directory **/usr/lpp/***pgm-name* does not exist.

inuupdt

The **inuupdt** command provides a common interface for applying an update to a single program. Normally, **updatep** runs **inuupdt**.

The **inuupdt** command has the following format:

inuupdt -d *device current-level new-level pgm-name*

where *pgm-name* is the name of a program and *current-level* specifies the current maintenance level. *new-level* is the level of the update to be applied. *pgm-name* can be a maximum of eight characters and *current-level* must be identical to the level value in the **/usr/lpp/***pgm-name***/lpp.hist** file.

The **inuupdt** command passes the following exit values to the process that called it:

0	Normal return.
2	Use the **sync** command to update the super blocks, i-nodes, and delayed block I/O, and then restart the VRM.
3	Build the kernel, then update the superblocks, i-nodes, and delayed block I/O (sync) and shut down the VRM.
4	Use the **cfgaply** subroutine to build the kernel. Use the **sync** command to update the super blocks, i-nodes, and delayed block I/O and then restart the VRM.
5	Installation cancelled without errors.
6	Update superblocks, i-nodes, and delayed block I/O (sync), then shut down the VRM.
7	Update cancelled by update procedure, recovery needed.
101-102	Places error code in the history file.
104-107	Places error code in the history file.
201-202	Places error code in the history file.
204-208	Places error code in the history file.

The **inuupdt** command returns the following exit status values:

100	Unknown return code received by **inuupdt**. It changes any unknown return code to 100 and logs it in the history file.
103	The restore of the archive file that contains the update control list, **/usr/lpp/***pgm-name***/inst_updt/arp**, failed.
201	An invalid flag was specified.
202	One or more parameters were missing.
203	Apply list does not exist or was not readable.
204	Too many parameters were entered.

Flag

-d *device* Updates the program from the specified *device*.

Files

/usr/include/inu21.h	Error code definitions for internal routines.
/usr/lpp/*pgm-name*/inst_updt	Temporary directory.
/usr/lpp/*pgm-name*/inst_updt.save	Directory for saved files.
/usr/lpp/*pgm-name*/inst_updt/arp	Program specific control library.
/usr/lpp/*pgm-name*/me.*vv.rr.llll*	Document change file.
/usr/lpp/*pgm-name*/ui.*vv.rr.llll*	Update instruction file.
/usr/sys/inst_updt	Temporary directory.
/usr/sys/inst_updt/control	Update control library.
/usr/sys/inst_updt/inutemp.*xx...x*	Temporary files.
/usr/sys/inst_updt/*pgm-name*_erata	Document change library.
/usr/sys/inst_updt/*pgm-name*_instr	Update instruction library.
/usr/sys/inst_updt/updt_cntrl	Temporary file.

Related Information

The following commands: "**installp**" on page 529 and "**bffcreate**" on page 108.

The discussion of code service in *Managing the AIX Operating System.*

The **lpp.hist** file in *AIX Operating System Technical Reference.*

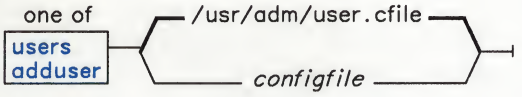
The discussion of updating programs in *AIX Operating System Programming Tools and Interfaces.*

users, adduser

Purpose

Adds, deletes, and changes user and group information.

Syntax



OL805076

Description

The **users** command lets you add, change, or delete user and group information in the **/etc/passwd**, **/etc/group**, **/etc/security/passwd**, and **/etc/security/group** files. To use the **users** command, you must be a member of the system group or have superuser authority (see "**su**" on page 1026).

The **users** command does all of its work in temporary files. When you enter the **q**uit subcommand, the temporary files become the permanent files. The old versions of **/etc/passwd** and **/etc/group** are renamed **/etc/opasswd** and **/etc/ogroup**. The old versions of **/etc/security/passwd** and **/etc/security/group** are renamed **/etc/security/opasswd** and **/etc/security/ogroup**. If **users** is ended by an **INTERRUPT (Alt-Pause)**, it removes the temporary files, and the system files remain as they were before the session. However, any directories created still exist, so you may have to remove directories after sending an **INTERRUPT**.

For configuration, **users** uses the file **/usr/adm/user.cfile**, the file specified with *configfile*, or the default parameters that follow:

Parameter	Default Value	Description
udir	/u/	Prefix of user home directory names.
program	null	The name of the user login program.
siteinfo	null	Any site-specific information.

Figure 12 (Part 1 of 2). Configuration File Parameters

Parameter	Default Value	Description
filesize	null	Size, in blocks, of the largest file that a user can make.
gname	staff	Name of the group to which a user is initially assigned.
minid	200	Minimum number that can be assigned as a user or group ID.
maxid	60000	Maximum number that can be assigned as a user or group ID.
pfile	/etc/passwd	Name of the password file.
gfile	/etc/group	Name of the group file.
owner	bin	Name of the owner of password and group files.
invalid	/usr/lib/sorry	Program for invalid accounts.

Figure 12 (Part 2 of 2). Configuration File Parameters

For information on how to use the **users** command, see *Managing the AIX Operating System.*

Notes:

1. The following files must exist on the same node:
   - **/etc/passwd**
   - **/etc/opasswd**
   - **/etc/security/passwd**
   - **/etc/security/opasswd**
   - **/etc/group**
   - **/etc/ogroup**
   - **/etc/security/group**
   - **/etc/security/ogroup**
   - **/usr/adm/user.cfile**
2. Each group has a limit of 500 users with eight-character IDs. Shorter IDs may allow more users per group.
3. It is possible to delete a user who still owns files or to delete a group that still has members. However, if you do this, it may cause problems later if the user name or group name is reused.

Subcommands

add	Adds a new user or group.
change	Changes data for an existing user or group.
delete	Deletes an existing user or group.
help	Displays a summary of available commands. Entering a question mark (?) also works for help.
invalidate	Changes a user's shell to a do-nothing program.
quit	Updates files and exits.
show	Shows information about a user or group.

The initial letter of each subcommand is recognized as the subcommand name.

Examples

The following is a sample **/usr/adm/user.cfile**:

```
pfile      /etc/passwd
gfile      /etc/group
owner      root
minid      200
maxid      1000
udir       /u/
program    /bin/sh
gname      staff
invalid    /usr/lib/sorry
```

Files

/usr/adm/user.cfile	Default configuration file.
/usr/adm/newuser.sys	Initialization shell file for added users.
/usr/adm/newuser.usr	Initialization shell file for added users.
/etc/passwd	Password file that identifies all known users.
/etc/security/passwd	A file which in conjunction with **/etc/passwd** contains information about user passwords.
/etc/group	Group file that identifies all known groups.
/etc/security/group	A file which in conjunction with **/etc/group** contains information about group passwords.
/etc/opasswd	Saved previous version of the password file.
/etc/security/opasswd	Saved previous version of the security password file.
/etc/ogroup	Saved previous version of the group file.
/etc/security/ogroup	Saved previous version of the security group file.

Related Information

The **group** and **passwd** files in *AIX Operating System Technical Reference.*

The discussion of users and passwords in *Managing the AIX Operating System.*

uucpadm

Purpose

Enters basic **uucp** configuration information

Syntax

A5AC5023

Description

The **uucpadm** command provides interactive entry and modification of basic **uucp** configuration information for the **Devices**, **Systems**, **Permissions**, **Poll**, and **Dialcodes** files.

The **uucpadm** command uses a copy of a file to record changes. The original file remains unchanged until you enter **Ctrl-U** or **Ctrl-X** at the appropriate menu. You can use **uucpadm** repeatedly to adjust the same file.

The **uucpadm** program checks the data as it is entered. If an inconsistency in the **uucp** files is found, **uucpadm** displays a warning message.

The help routine provides instructions for each data field. Enter a ? (question mark) at any menu component to access the help routine for that field. Enter a ~ (tilde) at any menu component to edit the appropriate file for that field. The **uucpadm** command invokes the editor designated by the **EDITOR** environmental variable. If **EDITOR** is not set, **uucpadm** invokes **/usr/bin/vi**.

If your entry for the first menu item matches an existing record, **uucpadm** retrieves that record for update. The **uucpadm** program also tells you how many records have that first entry.

Initial **uucp** configuration requires that local network information be included in the **Devices**, **Systems**, and **Permissions** files, in that order. The help routine provides examples of initial entries for these files.

The **Devices** file contains profiles of all physical devices which **uucp** can use to establish a connection with a remote host. The first field in the **Devices** file is *type*, used to indicate the type of communications device.

The following keywords are recognized by *type*:

Direct	A direct link to another computer.
ACU	An automatic call unit.
Built-in	A built in or standard function, such as **TCP**.
System name	A direct link to a specific computer.

The *line1* field specifies the device name associated with this entry. The *line2* field specifies the device name associated with a dialer. The *dialers* field is actually a dialer token pair field with the token optional. If the device is an automatic dialing modem, *dialers* is usually the brand name of the modem. In all cases the *dialers* field matches a record in the **Dialers** file.

A **Systems** file entry is required for each machine you communicate with. Unless *remote.unknown* is configured on your system, a **Systems** file entry is required for each machine that communicates with you. The *remote.unknown* entry allows access by unknown hosts.

The *name* field is the short node name of the remote host. The **uname -n** command can be used to obtain the *name* field. For example, if the **uname -n** command returns `lance.aus.ibm.com`, enter `lance` for *name*. The *time* field specifies when calls are permitted.

The **Permissions** file specifies options for machine access, file access, and command execution. These options take effect when a remote host logs in to your machine or you log in to a remote host.

The first prompt for an entry to the **Permissions** file is `L/M`. This entry must be either `LOGNAME=`(*login*) or `MACHINE=`(*short node name*). If `LOGNAME` is entered, permissions are defined which will take effect when a remote host enters this login ID. If `MACHINE` is entered, permissions are defined which take effect when you log in to a remote host.

The **Poll** file specifies machines in your **uucp** network to be polled. Each entry contains the name of the machine to be polled and the hours the machine should be called.

The **Dialcodes** file specifies dial code abbreviations to be used in the phone number entry of the **Systems** file. Each record contains an entry for abbreviation and dial code.

Japanese Language Support Information

If Japanese Language Support is installed on your system, this command is not available.

Examples

1. An entry to the **Devices** file with a `direct` 9600 baud connection to the `lance` machine on **/dev/tty2**:

```
Type = lance
line1 = tty2
line2 = -
class = 1200
dialers = direct
```

2. An entry to the **systems** file of the **lance.aus.ibm.com** system connected to an `ACU` device in `class` 2400:

```
Name = lance.aus.ibm.com
Time = Any
Type = ACU
Class = 2400
Phone = 997-7942
Login = nuucp
Password = saysme
```

3. A `LOGNAME` entry to the **Permissions** file:

```
L/M: LOGNAME=uucpz
Request: yes
Sendfiles: yes
Read: /
Write: NOWRITE=/etc
Callback:
Commands:
Validate: lance:backwoods
```

If the remote machine is `lance` or `backwoods`, the login ID must be `uucpz`. Remote hosts using this ID can `Request` to receive files, and your host can `Sendfiles` as requested. Users with this ID can read all files with *other* permissions and can write to all files, except those in **/etc**, with *other* permissions.

4. A MACHINE entry to the **Permissions** file:

```
L/M: MACHINE=lance
Request: yes
Sendfiles:
Read: NOREAD=/etc
Write: NOWRITE=/etc
Callback:
Commands: ALL
Validate:
```

   The machine ID is lance. Requests for file transfers can be made. The user can Read all files and can Write to all files except those in **/etc**. The execution of all Commands is permitted.

5. An entry to the **Poll** file consisting of the **lance.aus.ibm.com** system to be polled at 12pm, 9am, 1pm, and 6pm:

```
System = lance
Hours = 0 9 13 18
```

6. An entry to the **Dialcodes** file with LA as the abbreviation for the Los Angeles area code:

```
Abr = LA
Dialcode = 1-213-
```

Files

/usr/adm/uucp/Devices	Information about available devices.
/usr/adm/uucp/Dialcodes	Dialing code abbreviations.
/usr/adm/uucp/Dialers	Initial handshaking on a link.
/usr/adm/uucp/Permissions	Access permission codes.
/usr/adm/uucp/Poll	Machines to be polled.
/usr/adm/uucp/Systems	Accessible remote systems.

Related Information

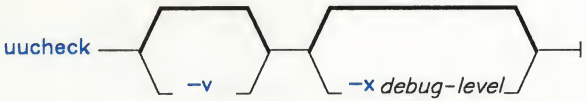
The following commands: "**uucp**" on page 1144, and "**uname**" on page 1114.

uucheck

Purpose

Checks for files and directories required by BNU.

Syntax



AJ2FL107

Description

The **uucheck** command checks for the presence of the files and directories required by the Basic Networking Utilities (BNU) facility. The command also checks for errors in the permissions file, **/usr/adm/uucp/Permissions**.

Note: The **uucheck** command does not check file/directory modes or some errors in the permissions file, such as duplicate login or machine names.

When BNU is installed, **uucheck** verifies that the directories, programs, and support files required to operate the networking facility are present. The command is executed automatically, as one of the first steps in the installation process, before the required BNU directories, programs, and files are actually installed.

Note: The **uucheck** command can be issued from the command line if the user has superuser privileges. For example, it would be useful to issue **uucheck** after making changes in part of the BNU facility such as the **Permissions** file.

Flags

-v	Gives a detailed explanation of how the BNU programs interpret the permissions file.
-x *debug_level*	Displays debugging information on the screen of the local terminal. The valid range for *debug_level* is 0 to 9. The higher the number, the more detailed the final report.

Files

/etc/locks/LCK*	Prevent multiple use of device.
/usr/adm/uucp/Devices	Information about available devices.
/usr/adm/uucp/Maxuuscheds	Limits scheduled jobs.
/usr/adm/uucp/Maxuuxqts	Limits remote command executions.
/usr/adm/uucp/Permissions	Access permission codes.
/usr/adm/uucp/Systems	Accessible remote systems.
/usr/spool/uucp/*	Spooling directory.
/usr/spool/uucppublic/*	Public directory.

Related Information

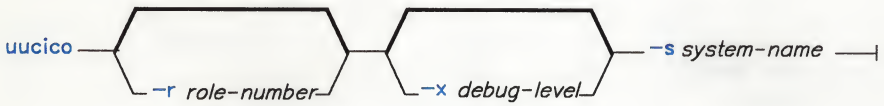
The following commands: "**uucico**" on page 1139, "**uusched**" on page 1156, "**uucp**" on page 1144, "**uustat**" on page 1158, and "**uux**" on page 1166.

uucico

Purpose

File transport program for the BNU facility.

Syntax



AJ2FL108

Description

The **uucico** program transports Basic Networking Utilities (BNU) requests. The BNU commands **uucp** and **uux** both queue jobs that are transferred to the specified computer by **uucico** after the required data, work, or execute files have been created.

The **uucico** program is normally started by the scheduler, **uusched**, but it can be started manually for debugging. The BNU commands **uutry**, **Uutry**, **Nutry**, and **uukick** also start **uucico** with debugging turned on.

The **uucico** program is a BNU daemon (a program executed internally to handle file transfers and command executions). It selects the device used for the communications link, establishes the connection to the remote computer, and performs the required login sequence. The program also performs permission checks, transfers data (**D.***) and command (**C.***) files, logs results, and notifies specified users of transfer requests.

Flags

-r *role_number* The role numbers are the number 1 for the server mode and the number 0 for client mode. The default is 0. If **uucico** is started manually, this flag should be set to 1.

-x *debug_level* Displays debugging information on the screen of the local terminal. The valid range for *debug_level* is 0 to 9. The higher the number, the more detailed the final report. This flag is useful in correcting problems with the *expect-send* sequence in the **Systems** file.

-s *system_name*
The name of the remote system. Use only when starting **uucico** manually. The *system_name* is supplied internally when **uucico** is started automatically. System names must contain only ASCII characters.

Files

/etc/locks/LCK*	Prevents multiple use of device.
/usr/adm/uucp/Devices	Information about available devices.
/usr/adm/uucp/Dialcodes	Dialing code abbreviations.
/usr/adm/uucp/Dialers	Initial handshaking on a link.
/usr/adm/uucp/Maxuuscheds	Limits scheduled jobs.
/usr/adm/uucp/Maxuuxqts	Limits remote command executions.
/usr/adm/uucp/Permissions	Access permission codes.
/usr/adm/uucp/Systems	Accessible remote systems.
/usr/spool/uucp/*	Spooling directory.
/usr/spool/uucppublic/*	Public directory.

Related Information

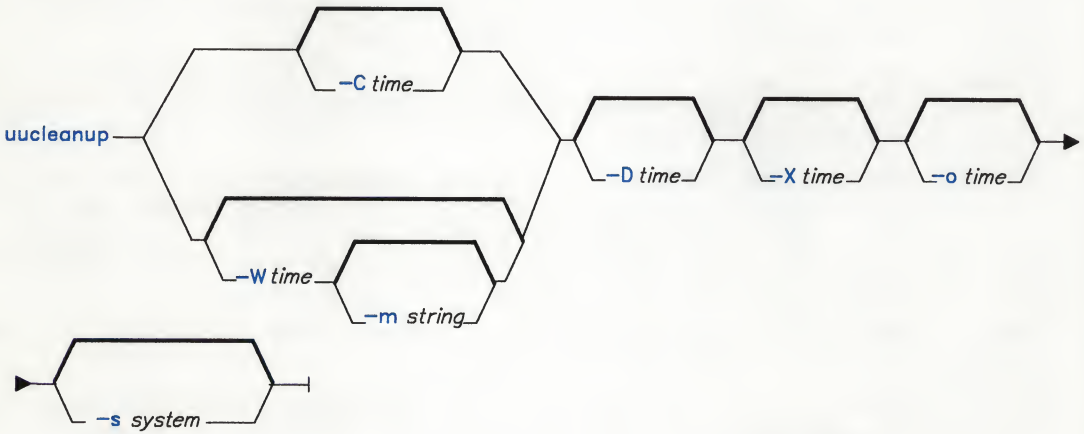
The following commands: "**cron**" on page 220, "**uucp**" on page 1144, "**uusched**" on page 1156, "**uustat**" on page 1158, "**uutry, Uutry, uukick**" on page 1164, and "**uux**" on page 1166.

uucleanup

Purpose

Deletes selected files older than a specified number of hours from the BNU spool directory or a named directory.

Syntax



AJ2FL109

Description

The Basic Networking Utilities (BNU) program **uucleanup** scans the spool directory (**/usr/spool/uucp**) for old files and takes appropriate action to remove them in a useful way. Used primarily by the BNU program administrator, **uucleanup** performs the following tasks:

- Informs the requester of send/receive requests for systems that cannot be reached
- Warns users about requests that have been waiting for a given number of days; the default is 1 day
- Returns mail that cannot be delivered to the sender
- Removes all other files older than a specified number of days from the spool directory.

The **uucleanup** program is started by the shell **uudemon.cleanup**, located in **/usr/adm/uucp**, which in turn is started by the **cron** script, located in **/usr/spool/cron/crontabs/uucp**. In general, **uucleanup** is executed automatically. You can also start the **uucleanup** program manually if you have superuser privileges.

Note: When BNU is installed, automatic cleanup is not enabled. Edit the file **/usr/spool/cron/crontabs/uucp** and remove the comment character "#" from the beginning of the **uudemon.cleanup** line.

Flags

-C*time* Removes any **C.*** (command) files as old as, or older than, the number of days specified in *time*, and sends appropriate information to the requester. Unless specified otherwise, the default *time* is 7 days.

-D*time* Removes any **D.*** (data) files as old as, or older than, the number of days specified in *time*. Also attempts to deliver any remaining mail messages. The default *time* is 7 days.

-W*time* Sends a mail message to the requester warning that **C.** files as old as, or older than, the number of days specified in *time* are still in the spool directory. The message includes the job ID and, in the case of mail, the mail message. The administrator may use the **-m** option to include a message line telling whom to call to check the problem. The default *time* is 1 day.

-X*time* Removes any **X.*** (execute) files as old as, or older than, the number of days specified in *time*. The default *time* is 2 days.

Note: There are probably no related data files. If any related data files remain, however, they are handled by **D.*** processing, as described above.

-m*string* Includes a specified line of text in the warning message generated by the **-W***time* option. The default line is: `See your local administrator to locate the problem.`

-o*time* Removes other files as old as, or older than, the number of days specified in *time*. The default *time* is 2 days.

-s*system* Executes **uucleanup** only on the spool directory specified by *system* The default is to clean up all BNU spool directories. System names can contain only ASCII characters.

Note: Unless one of the *time* flags is set to a specific number of days, **uucleanup** uses the default *times* values.

Files

/etc/cron	File that starts **uudemon.cleanup**.
/usr/adm/uucp	Directory with commands used internally by **uucleanup**.
/usr/spool/cron/crontabs/uucp	File containing **uudemon.cleanup**.
/usr/spool/uucp	Spooling directory.

Related Information

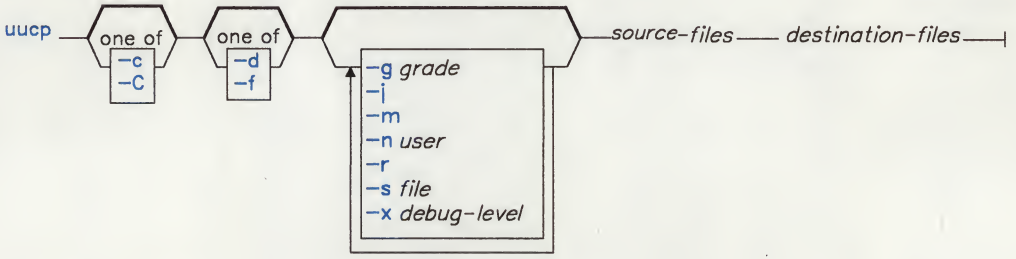
The following commands: "**cron**" on page 220, "**uucp**" on page 1144, and "**uux**" on page 1166.

uucp

Purpose

Copies files from one AIX system to another AIX system.

Syntax



OL805382

Description

The Basic Networking Utilities (BNU) command **uucp** copies one or more *source* files from one AIX system to one or more *destination* files on another AIX system.

The **uucp** command accomplishes the file transfer in two steps: first, by creating a command (**C.***) file in the spooling directory on the local computer, and then by sending the request to the specified computer via the **uucico** command.

Command files include information such as the full path name of the source and destination files, the sender's login name, and so on. The full path name of a command file is a form of the following:

/usr/spool/uucp/*system_name*/**C.***system_nameNxxxx*

where *N* is the grade of the request and *xxxx* is the hexadecimal sequence number used by BNU.

Note: If the **uucp** command is used with the **-C** flag to copy the files to the spool directory for transfer, **uucp** creates not only a command file, but also a data (**D.***) file that contains the actual source file. The full path name of a data file is a form of the following:

/usr/spool/uucp/*system_name*/**D.***system_namexxxx###*

Once the command files (and data files, if necessary) are created, **uucp** then calls the **uucico** daemon, which in turn attempts to contact the remote computer to deliver the files.

Note: It is useful to issue the **uuname** command to determine the exact name of the remote system before issuing **uucp**. The **uulog** command provides information about **uucp** activities on a system.

Path Names Used with uucp

Path names for the source and destination of the **uucp** transfer may be one of the following:

- A full path name
- A relative path name
- A path name preceded by ~*user*, where *user* is a login name on the specified system. The specified user's login directory is then considered the destination of the transfer.

  If the user specifies an invalid login name, the files are transferred to the public directory, **/usr/spool/uucppublic**, which is the default.
- A path name preceded by ~/*destination*, where *destination* is appended to **/usr/spool/uucppublic**.

  This destination is treated as a file name unless more than one file is being transferred by this request, or the destination is a directory. To ensure that it is a directory, follow the destination name with a / (slash). For example, `~/amy/` as the destination creates the directory `/user/spool/uucppublic/amy`, if it does not already exist, and puts the requested files in that directory.

Note: Path names can contain only ASCII characters.

Source and Destination File Names

- A file name can be a path name on the local system, or can have the following form:

  system_name!*path_name*

  where *system_name* is taken from a list of system names that BNU knows about.
- The destination *system_name* can also be a list of names, such as the following:

  system_name!*system_name*! . . . ! *system_name*!*path_name*

  In this case, an attempt is made to send the file via the specified route to the destination. Make sure that intermediate nodes in this route are willing to forward information (see *Managing the AIX Operating System*).
- The shell pattern-matching characters ?, *, and [. . .] may be used in the path names; the appropriate system expands them.

  Note: The shell pattern-matching characters should not be used in the path name of the destination file.

- If the *destination* is a directory rather than a file, **uucp** uses the last part of the *source* name.

Permissions

- The system administrator should restrict the access to local files by users on other systems.
- When transmitting files, **uucp** preserves execute permissions and grants read and write permissions to the owner, the group, and all others. (The **uucp** command owns the file.)
- Sending files to arbitrary *destination* path names on other systems, or getting files from arbitrary *source* path names on other systems, often fails because of security restrictions. The files specified in the path name must give read or write permission not only for the same group of users, but also for any group.
- Protected files and files in protected directories owned by the requestor can be sent by **uucp**.

Note: File names and system names can contain only ASCII characters.

Flags

-c Transfers the source files to the destination on the specified computer. The source files are not transferred via the spool directory. This saves the system from copying possibly large files to the spooling directory for transfer. (See the discussion of the **-C** flag.) This flag is on by default.

-C Copies local files to the spool directory for transfer. Depending on the configuration of the **Poll** and **Systems** files, and on how often the **uusched** command is run, the files could be transferred immediately (on demand polling), or in the future.

Note: Occasionally, there are problems in transferring a source file; for example, the remote computer may not be working, or the login attempt may fail. In such a case, the file remains in the spool directory until it is either transferred successfully or removed by the **uucleanup** command.

-d Creates any intermediate directories needed to copy the source files to the destination. This flag is on by default.

-f Does not create intermediate directories during the file transfer.

-g*grade* Specifies when the files are to be transmitted during a particular connection. *Grade* is a single number (0-9) or letter (A-Z, a-z); lower ASCII-sequence characters cause the files to be transmitted earlier than do higher sequence characters. The number 0 is the highest (earliest) grade; z is the lowest (latest) grade. The default is **N**.

-j	Displays the job identification number of the transfer operation on standard output. This job ID can be used by the BNU command **uustat** to obtain the status of a information about the status of a particular job, or with **uustat -k** to terminate the transfer before it is completed.
-m	Sends mail to the requester when the transfer to the remote system is completed. The message is sent to the requester's mailbox, **/usr/mail/***user_name*. The **mail** command does not send a message for a local transfer. **Note:** The **-m** flag works only when sending files or receiving a single file. It does not work when forwarding files. Receiving multiple files specified by the shell pattern-matching characters ?, *, and [. . .] does not activate the **-m** option.
-n*user_name*	Notifies the user specified by *user_name* on the designated system that files have been sent. The mail system does not send a message for a local transfer. **Note:** User names can contain only ASCII characters.
-r	Prevents the starting of the file transfer program, **uucico**, even if the command was issued at a time when calls to the remote system are permitted. By default, a call to the remote system is attempted if the command is issued during a time period specified in the **Poll** and **Systems** files.
-s*file*	Reports the status of the transfer to the specified file. In this case, the *file* designation must be a full path name.
-x*debug_level*	Displays debugging information on the screen of the local system. The *debug_level* is a number between 0 and 9. The higher number gives a more detailed report.

Examples

1. To copy one or more files locally, within the same directory:

```
uucp file1 file2
```

2. To copy multiple files locally, from one directory to another directory:

```
uucp /dev/geo/project /usr/test/marg
```

3. To copy file `f1` from the local system to a remote system named `hera`:

```
uucp /u/geo/f1 hera!/u/geo/f1
```

4. To copy file `f2` from the remote system `hera` and place it in the public directory:

```
uucp hera!geo/f2 /usr/spool/uucppublic/f2
```

5. To place the f2 file in a directory other than the public directory:

   ```
   uucp hera!geo/f2 /u/geo/f2
   ```

   In this case, make sure that the geo login directory allows write permission to both "other" user and "other" group (for example, with mode 777).

Files

/usr/spool/uucp	Spooling directory.
/usr/spool/uucppublic	Public directory.
/usr/lib/uucp	Contains **uucico** daemon.

Related Information

The following commands: "**mail, Mail**" on page 608, "**uucleanup**" on page 1141, "**uulog**" on page 1149, "**uuname**" on page 1151, "**uusched**" on page 1156, "**uustat**" on page 1158, "**uux**" on page 1166, and "**uuxqt**" on page 1172.

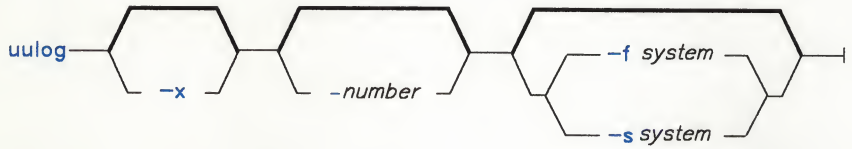
The information about international character support in *Managing the AIX Operating System*.

uulog

Purpose

Provides information about **uucp** and **uux** activities on a system.

Syntax



AJ2FL111

Description

The Basic Networking Utilities (BNU) command **uulog** displays the contents of a log file of **uucico** or **uuxqt** activities. Individual log files are created for each remote system with which the local system communicates using the **uucp**, **uuto**, or **uux** commands.

The log file of **uucico** activities is named **/usr/spool/uucp/.Log/uucico/***system*. The log file of **uuxqt** activities is named **/usr/spool/uucp/.Log/uuxqt/***system*.

Flags

-f*system*	Performs a "tail **-f**" on the file transfer log for the specified *system*, in this case displaying the end of the log file. Use **INTERRUPT** (**Alt-Pause**) to leave the file and return to the prompt.
-s*system*	Prints information about copy requests involving the specified *system*. **Note:** System names can contain only ASCII characters.
-x	Looks in the **uuxqt** log file for the given system.
-number	Indicates that a **tail** command should be executed for the specified *number* of lines.

Files

/usr/bin	Contains **uulog** command.
/usr/spool/uucp	Spooling directory.
/usr/spool/uucppublic	Public directory.

Related Information

The following commands: "**tail**" on page 1044, "**uucp**" on page 1144, "**uuname**" on page 1151, and "**uux**" on page 1166.

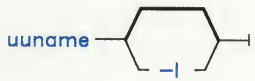
The information about international character support in *Managing the AIX Operating System*

uuname

Purpose

Provides information about other systems accessible to the local system.

Syntax



AJ2FL112

Description

The Basic Networking Utilities (BNU) command **uuname** displays a list of all the computers networked to the local system; the list of accessible systems is displayed on the screen of the local terminal.

In order for a local system to communicate with a remote system via BNU, the remote system must:

- have a UNIX-based operating system
- be connected to the local system.

Note: BNU can be used to communicate between an RT work station and a non-UNIX-based operating system, but such communications may require special hardware or software. The remote systems accessible with BNU commands are identified when the BNU programs are installed, and are listed in **/usr/adm/uucp/Systems**.

Before copying a file to another system with the **uucp** command, issue **uuname** to determine the exact name of the remote system.

Flags

-l Displays the name of the local system.

Examples

1. To identify the remote systems connected to the local systems:

   ```
   uuname
   ```

   The system responds with a list like the following:

   ```
   hera
   zeus
   merlin
   arthur
   ```

2. To identify the local system:

   ```
   uuname -l
   ```

   The system responds:

   ```
   venus
   ```

Files

/usr/spool/uucp	Spooling directory.
/usr/spool/uucppublic	Public directory.
/usr/adm/uucp	Directory containing **Systems** file.

Related Information

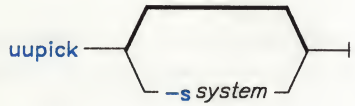
The following commands: "**uucp**" on page 1144, "**uulog**" on page 1149, and "**uux**" on page 1166.

uupick

Purpose

Accepts or rejects files transmitted to a user.

Syntax



A5AC5019

Description

The Basic Networking Utilities (BNU) command **uupick** accepts or rejects files that the BNU command **uuto** has transmitted to a designated user **ID**.

After the files have arrived, the **rmail** command notifies the specified user. At that point, the user issues **uupick** to receive and handle the files.

Specifically, **uupick** searches the public directory on the local system for files with some form of the following name:

/usr/spool/uucppublic/receive/*user_ID*/*system*/*file*

For each entry (file or directory) found, **uupick** displays the following message on the screen of the local system:

`from` *system*`: [file` *file-name*`] [dir` *dirname*`]`
`?`

It then waits for a response from standard input to determine the disposition of the file. Issuing the **uupick** command with the appropriate file-handling option completes the transfer.

File-Handling Options

After notifying the specified user that a file has been sent from *system*, **uupick** displays a question mark (?), prompting for one of the following file-handling options:

*	Displays all the file-handling options.
Enter	Moves on to the next entry in the **receive** directory.
a [*dir*]	Moves all **uuto** files currently in the **receive** directory into a specified directory on the local system. The default is the current working directory. Use a full or relative path name to specify *dir*.
d	Deletes the specified file.
m [*dir*]	Moves the specified file to a specified directory. If *dir* is not specified as a complete path name, a destination relative to the current directory is assumed. If no destination is given, the default is the current working directory on the local system.
p	Displays the contents of the file on the work station screen.
q	Stops processing and exits from the **uupick** command.
Ctrl-D	Same as **q**.
!*cmd*	Escapes to a shell to run the specified AIX command. After the command executes, returns automatically to **uupick** so the user can continue to handle the **uuto** files in the **receive** directory.

Flags

-s*system*	Searches **/usr/spool/uucppublic/receive/***user_ID*/*system*/*file* only for files sent from the specified system. **Note:** System names can contain only ASCII characters.

Examples

1. To receive `file1` sent with the **uuto** command from user `msg` on system `apollo`:

   ```
   uupick
   ```

   The system responds:

   ```
   from system apollo: file file1
   ?
   ```

2. Enter an asterisk (*) to display the **uupick** file-handling options:

```
?
*
```

The system responds:

```
usage [d] [m dir] [a dir] [p] [q] [cntl d] [!cmd] [*] [new-line]
```

Enter the appropriate option, or use the **q** option or the **Ctrl-D** sequence to exit from the **uupick** command.

Files

/usr/spool/uucppublic	Public directory.

Related Information

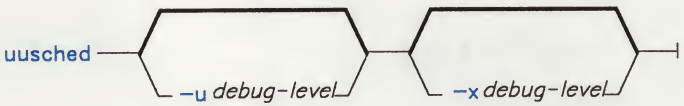
The following commands: "**bellmail**" on page 104, "**uuto**" on page 1162, "**uucp**" on page 1144, and "**uux**" on page 1166.

uusched

Purpose

Schedules work for the BNU file transport program.

Syntax



AJ2FL113

Description

The **uusched** program is the Basic Networking Utilities (BNU) file-transport scheduler. It is one of the BNU daemons (a program executed internally to handle file transfers and command executions).

The **uusched** daemon schedules the transfer of files that are queued in the **/usr/spool/uucp** directory. The scheduling program first randomizes the work and then starts the **uucico** daemon with the **-s** option. This option specifies the computer for which the particular job is scheduled.

The **uusched** program itself is usually started by the shell **uudemon.hour**, which is started from **/usr/spool/cron/crontabs/uucp**, which is, in turn, started by **cron**.

Flags

-u*debug_level* Passes as **-x***debug_level* to **uucico**. The *debug_level* is a number from 0 to 9. Higher numbers give more detailed debugging information, which is displayed on the screen of the local system.

-x*debug_level* Outputs debugging messages from **uusched**. The *debug_level* is a number from 0 to 9. Higher numbers give more detailed debugging information.

Files

/etc/locks/LCK*	Prevents multiple use of device.
/usr/adm/uucp/Devices	Information about available devices.
/usr/adm/uucp/Dialcodes	Dialing code abbreviations.
/usr/adm/uucp/Dialers	Initial handshaking on a link.

/usr/adm/uucp/Permissions	Access permission codes.
/usr/adm/uucp/Systems	Accessible remote systems.
/usr/spool/cron/crontabs/uucp	Contains **uudemon.cleanup**.
/usr/spool/uucp/*	Spooling directory.
/usr/spool/uucppublic/*	Public directory.

Related Information

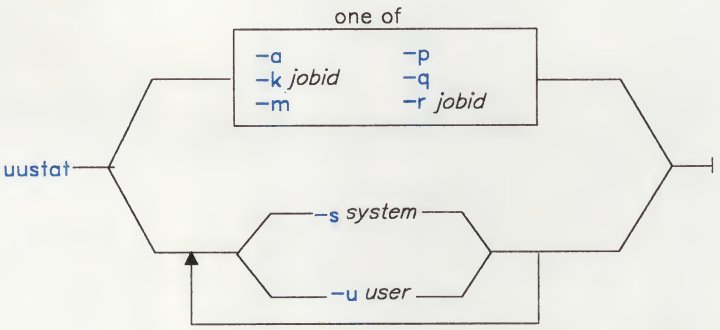
The following commands: "**cron**" on page 220, "**uucico**" on page 1139, "**uucp**" on page 1144, "**uustat**" on page 1158, and "**uux**" on page 1166.

uustat

Purpose

Reports the status of and provides rudimentary job control for BNU commands.

Syntax



AJ2FL114

Description

The Basic Networking Utilities (BNU) command **uustat** displays status information about several types of BNU operations. It is particularly useful in monitoring transfer (copy) requests issued with the **uucp** and **uuto** commands, and requests to run an AIX command(s) on a remote system made with the **uux** command.

In addition, **uustat** also gives a user limited control over BNU jobs queued to run on remote systems. By issuing the command with the appropriate flag, a user can check the general status of BNU connections to other systems, and cancel copy requests made with **uucp** and **uuto**.

If **uustat** is issued without any flags, the command reports the status of all BNU requests issued by the current user since the last time the holding queue was cleaned up (see the description of the **-a** flag for an explanation of the BNU queues). Such status reports are displayed in the following format:

jobid date/time status system_name user_ID size file

See "Examples" on page 1160 for an explanation of this format.

Note: When sending files to a system that has not been contacted recently, it is a good idea to use **uustat** to see when the last access occurred, as the remote system may be down or out of service.

Flags

The following flags are mutually exclusive; you can use only one at a time with the **uustat** command:

-a Displays information about all the jobs in the holding queue, regardless of the user who issued the original BNU command.

Note: There are two types of BNU queues.

- The current queue lists the BNU jobs either queued to run on, or currently executing on, one or more specified computers. Use the **uustat -q** command to examine this queue.
- The holding queue, accessed with the **-a** flag, lists all jobs that have not executed during a set period of time.

  After the set time period has elapsed, the entries in the holding queue are deleted either manually, with the BNU command **uucleanup**, or automatically, with the file **/usr/spool/cron/crontabs/uucp** (which includes **uudemon.cleanup**), which is started by **cron**.

-k *jobid* Cancels (kills) the BNU process specified by the *jobid*. The person using this flag must either be the one who made the **uucp** request now being canceled, or must be operating with superuser authority.

Note: This flag cancels a process only when that job is still on the local computer. Once BNU has moved the job to a remote system for execution, **-k***jobid* cannot be used to cancel the remote job.

-m Reports the status of the most recent attempt to contact the specified system with a BNU command. If the BNU request was completed, the status report is SUCCESSFUL. If the job was not completed, the status report is an error message such as LOGIN FAILED.

-p Runs a **ps -flp** (process status: full, long list of specified process IDs) for all PID numbers in the lock files.

-q Lists the jobs currently queued to run on each system; these jobs are either waiting to execute or in the process of executing. If a status file exists for the system, its date, time, and status information are reported. Once the job is finished, BNU removes that job listing from the current queue.

Note: In a status report, a number in parentheses next to the number of a **C.*** (command) file or an **X.*** (execute) file represents the age in days of the oldest **C.*/X.*** file for that system. The retry field represents the number of times BNU tried and failed to execute the command because of such factors as a failed login, locked files, an unavailable device, and so on.

-r *jobid* Marks the files in the holding queue specified by *jobid* with the current date and time. Use this flag to ensure that a cleanup operation does not delete files until the job's modification time reaches the end of the specified period.

You can use either one or both of the following flags with **uustat**:

-s*system* Reports the status of BNU requests for the work station specified by *system*.

-u*user_ID* Reports the status of BNU requests by the specified user for any work station.

Note: System and user names can contain only ASCII characters.

Examples

1. To display the status of all BNU jobs in the holding queue:

   ```
   uustat -a
   ```

   The system responds with a display like the following:

   ```
   heraC3113   11/06-17:47 S hera   amy 289   D.venus471afd8
   zeusN3130   11/06-09:14 R zeus   geo 338   D.venus471bc0a
   merlinC3120 11/05-16:02 S merlin amy 828   /u/amy/tt
   merlinC3119 11/05-12:32 S merlin msg rmail amy
   ```

   The first field is the job ID of the operation, which is followed by the date and time the BNU command was issued. The third field is either an S or an R, depending on whether the job is to send or request a file. The fourth field is the name of the system on which the command was entered, followed by the user ID of the person who issued the command. The sixth field is the size of the file, or, in the case of a remote execution like the last entry in the example, the name of the remote command. When the size is given, as in the first three lines of the example output, the file name is also displayed. The file name can be either the name given by the user, as in the `/u/amy/tt` entry, or a name that BNU assigns internally to data files associated with remote executions, such as `D.venus471afd8`.

2. To display the status of all jobs in the current queue:

   ```
   uustat -q
   ```

   The system responds:

   ```
   merlin  3C      07/15-11:02  NO DEVICES AVAILABLE
   hera    2C      07/15-10:55  SUCCESSFUL
   zeus    1C (2)  07/15-10:59  CAN'T ACCESS DEVICE
   ```

   The output tells how many **C.*** (command) files are waiting for each system. The date and time refer to the current interaction with the system, followed by a report of the status of the interaction. The number in parentheses (2) in the third line of the example indicates that the **C.*** file has been in the queue for two days.

3. To display all process IDs in the lock file:

```
uustat -p
LCK..tty0: 881
LCK.S.0: 879
LCK..hera: 881
F   S UID  PID PPID C  PRI NI ADDR SZ  WCHAN    STIME    TTY
101 S uucp 881 879  26 39  39 370  296 3fffe800 09:57:03  -
TIME   COMD
0:00   UUCICO -rl -shera
101 S uuc  879 1    11 33  39 770  156 8d874     09:57:02  -
0:00   /usr/lib/uucp/uusched
```

4. To cancel a job in the current queue, first determine the job ID and then execute the **uustat -k** command:

```
uustat -a
heraC3113   11/06-17:47 S hera   amy 289 D.venus471afd8
merlinC3119 11/06-17:49 S merlin geo 338 D.venus471bc0a
uustat -k heraC3113
```

5. To report the status of jobs requested by system `hera`:

```
uustat -s hera
heraN1bd7  07/15-12:09 S hera amy 522    /user/amy/A
heraC1bd8  07/15-12:10 S hera amy  59    D.3b2a12ce4924
heraC3119  07/15-12:11 S hera amy rmail  msg
```

6. To report the status of jobs requested by user `amy`:

```
uustat -u amy
```

This flag displays output similar to that produced by the **-s** flag.

Files

/etc/locks/LCK*	Prevents multiple use of device.
/usr/spool/uucp	Spooling directory.

Related Information

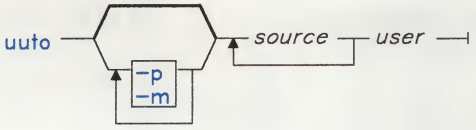
The following commands: "**ps**" on page 786, "**uucp**" on page 1144, "**uuto**" on page 1162, and "**uux**" on page 1166.

uuto

Purpose

Copies public files from one AIX system to another AIX system, with local system control of file access.

Syntax



OL805290

Description

The Basic Networking Utilities (BNU) command **uuto** copies one or more *source* files from one AIX system to a specified user on another AIX system. The **uuto** command calls the BNU command **uucp** for the actual file transfer, but **uuto** enables the recipient to use the **uupick** options to handle the transferred files on the local system.

The *source* entry is the name of the files on the local system, or a path name to the files on the system that runs the command. The *user* is a specific user ID. This entry has the following format:

system!*user*

where *system* is the name of a remote system connected to the local system, and *user* is the login name of the recipient of the transferred files on the specified system.

Note: When copying a file from one user to another user on the local system, omit the *system* entry; the destination is simply the ID of the user to whom the file is being sent. System names can contain only ASCII characters.

The **uuto** command sends files to **/usr/spool/uucppublic** on the designated *system*; this is a public directory. The command also creates an additional directory called **receive** (if it does not already exist), plus the directory */user/system*. The full path names to the copied files are therefore some form of the following:

/usr/spool/uucppublic/receive/*user*/*system*/*files*

Once the copied file is in the **receive** directory, **uuto** notifies the recipient by **rmail** that the file has arrived. The recipient then issues the **uupick** command, which searches the public directory for files sent to the specified user ID, displaying the message that file *name* has arrived from system *name* for each file it locates. The user then enters one of the **uupick** file-handling options to delete the file, move it to another directory, and so on.

Flags

-m Notifies the sender by **bellmail** when the copy is complete.

-p Copies the source file to the spool directory on the local system. The source file resides in the spooling directory for a set period of time (defined in the **uusched** program) before the **uucp** command calls the **uucico** daemon, which actually transfers the copy to the public directory on the specified remote system. The default is to transfer a source file directly to the specified user.

Files

/usr/spool/uucppublic	Public directory.

Related Information

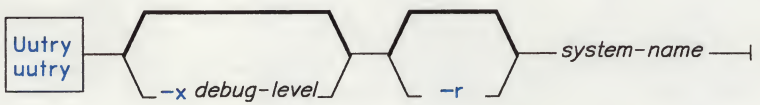
The following commands: "**bellmail**" on page 104, "**uucleanup**" on page 1141, "**uucp**" on page 1144, "**uupick**" on page 1153, "**uusched**" on page 1156, "**uustat**" on page 1158, and "**uux**" on page 1166.

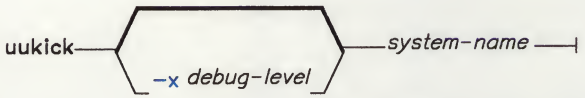
uutry, Uutry, uukick

Purpose

Contacts a remote system with debugging turned on.

Syntax





AJ2FL258

Description

The **uutry** command contacts a specified system with debugging turned on, thus providing a means of checking call processing capabilities with debugging output. This command invokes the **uucico** program, which in turn establishes the actual connection to the remote system.

The debugging output (information about the progress of **uucico** in establishing the connection, performing the remote login, and so on) is scrolled on the screen of the local system. Once the system has finished displaying this information, use **INTERRUPT (Alt-Pause)** to return to the prompt.

The debugging information scrolls rapidly, so you may want to direct that output to a file by issuing **Uutry** rather than **uutry**.

The **Uutry** command (note the uppercase "U") works almost exactly like **uutry**, with one exception. In addition to displaying the debugging output on the screen, **Uutry** also directs this information to a file named **/tmp/***system_name*. Again, when the last of the output has been displayed, use **INTERRUPT** to return to the prompt.

Note: You can also press **INTERRUPT** while the system is scrolling the output generated by **Uutry**. This returns you to the prompt, while the **uucico** program continues to place the debugging information in **/tmp/***system_name*. Use the **pg** command to examine this file.

Note: System names can contain only ASCII characters.

The **uukick** command also works just like the **uutry** command. The only difference between the two commands is that **uukick** takes only the **-x***debug_level* flag. You cannot override the retry time with the **-r***system_name* flag.

Flags

-x*debug_level*	Used in debugging to override the default level of 5, and produce a detailed output of the program execution. The debugging level is a single digit between 0 and 9. Higher numbers produce more detailed debugging information, which is displayed on the screen of the local system.
-r	Overrides the retry time specified in **/usr/spool/uucp/.Status**.

Files

/etc/locks/LCK*	Prevents multiple use of device.
/tmp/*system_name*	Temporary data file.
/usr/adm/uucp/Devices	Information about available devices.
/usr/adm/uucp/Dialcodes	Dialing code abbreviations.
/usr/adm/uucp/Dialers	Initial handshaking on a link.
/usr/adm/uucp/Masuuscheds	Limits scheduled jobs.
/usr/adm/uucp/Masuuxqts	Limits remote command executions.
/usr/adm/uucp/Permissions	Access permission codes.
/usr/adm/uucp/Systems	Accessible remote systems.
/usr/spool/uucp/*	Spooling directory.
/usr/spool/uucppublic/*	Public directory.

Related Information

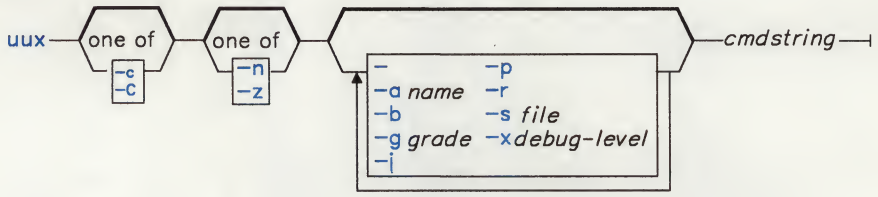
The following commands: "**uucico**" on page 1139, "**uucp**" on page 1144, and "**uux**" on page 1166.

uux

Purpose

Runs a command on another AIX system.

Syntax



AJ2FL116

Description

The Basic Networking Utilities (BNU) command **uux** runs a specified AIX command on a specified AIX system.

The command gathers various files from the designated systems, if necessary. It then runs a specified command on a designated system. The user can direct the output from the command to a specified file on a specified system.

Note: For security reasons, many installations permit **uux** to run only the **rmail** command.

The **uux** command creates execute (**X.***) files that run AIX commands on the local system. In addition, **uux** also creates both command (**C.***) files and data (**D.***) files. Execute files contain the command string to be executed on the designated system. Command files contain the same information as those created by the **uucp** command. Data files either contain the data for a remote command execution, or else become **X.*** files on remote systems for remote command executions.

Note: The full path name of an execute file is a form of the following:

/usr/spool/uucp/*system_name*/*X.system_nameNxxxx*

After creating the files in the spooling directory, **uux** calls the **uucico** daemon, which in turn attempts to contact the designated system to deliver the files. Once the files are transferred, the **uuxqt** daemon executes the *cmdstring* on the specified system.

The *cmdstring* is made up of one or more arguments that look like an AIX command line, except that *cmdstring* may be prefixed by *system_name*!. The default *system_name* is the local system.

Note: To run commands on more than one system, type the information on separate command lines:

```
uux merlin!print /reports/memos/charles
uux zeus!print /test/examples/exampl
```

Unless the **-n** or **-z** flag is specified, **uux** notifies the user if the remote system fails to run the command. The response comes by **mail** from the other system.

File Names, Path Names, and System Names

- When specifying the destination of the output of a command, **uux** may be entered in either one of the following formats:
  - **uux** [*options*] *"cmdstring* > *destination_name"*
  - **uux** [*options*] *cmdstring* \{*destination_name*\}
- Destination names may be either of the following:
  - a full path name.
  - a full path name preceded by ~*user*, where *user* is a login name on the specified system. The **uux** command replaces this path name with the user's login directory.
- The shell pattern-matching characters ?, *, and [. . .] can be used in the path name of a "source" file (such as files compared by the **diff** command); the appropriate system expands them. However, using the * character may occasionally produce unpredictable or unanticipated results.

  Note: Shell pattern-matching characters should not be used in the destination path name.
- Place either two backslashes (\ . . . \) or a pair of quotation marks (" . . . ") around pattern-matching characters in a path name so the local shell cannot interpret them before **uux** sends the command to a designated system.
- If using the special shell characters > (greater than), < (less than), ; (semicolon), or | (vertical bar) in a path name, place either \ . . . \ or " . . . " around the individual character or around the entire command string.
- Do not use the shell redirection characters < < or > > in a path name.
- The **uux** command attempts to move all files specified on the command line to the designated system. Enclose the names of all output files in parentheses so that **uux** does not try to transfer them.
- When specifying a *system_name*, always place it before the *cmdstring* in the entry.

Note: System names can contain only ASCII characters.

- The exclamation point preceding the name of the local system in a command is optional. If you choose to include the ! to run a command on the local system using files from two different remote systems, use ! instead of *system*! to represent the local system, and add *system*! as the first entry in any path name on the remote systems.
- The exclamation point representing a remote system in BNU syntax has a different meaning in C shells (**csh**). When running **uux** in a C shell, place a backslash (\) before the exclamation point in a system name.
- If the command being executed requests two files stored on the same system, or two files with the same name that are stored on separate systems, the command will execute, but will not produce the desired results.

  The following two commands *will* execute:

```
uux "hera!/bin/diff /usr/amy/out1 hera!/u/amy/out > ~uucp/DF"
uux "hera!/bin/diff hera!/usr/amy/out1 venus!/u/amy/out > ~uucp/DF"
```

  Note: The notation `~uucp` is the shorthand way of specifying the public spooling directory **/usr/spool/uucppublic**.

  In the first command, `diff` is on system `hera`, the first source file is on the local system, the second source file (with a different name) is on system `hera`, and the output is directed to the file `DF` in the public directory on the local system. In the second command, `diff` is again on `hera`, the first file is also on `hera`, the second file (with a different name) is on `venus`, and the output is again directed to `DF` in the `~uucp` directory.

  The following command will not execute properly:

```
uux "hera!/bin/diff venus!/u/amy/out merlin!/u/amy/out > ~uucp/DF"
```

  This command will not execute because, although the files are on two different systems, they still have the same file name.

Flags

-	Makes the standard input to **uux** the standard input to the *cmdstring*.
-a*name*	Replaces the user ID of the person issuing the command with user ID specified with *name*.
-b	Returns standard input to the command if the exit status is not zero.

-c Transfers the source files to the destination on the specified system. The source files are not copied into the spool directory for transfer. (See the discussion of the **-C** flag.) This flag is on by default.

-C Transfers the source files to the spool directory. After a set period of time (specified in the **uusched** program), the **uucico** daemon attempts to transfer the files to the destination on the specified computer.

Note: Occasionally, there are problems in transferring a source file; for example, the remote computer may not be working, or the login attempt may fail. In such cases, the file remains in the spool directory until it is either transferred successfully or removed by the **uucleanup** command.

-g*grade* Specifies when the files are to be transmitted during a particular connection. *Grade* is a single number (0-9) or letter (A-Z, a-z); lower ASCII-sequence characters cause the files to be transmitted earlier than do higher sequence characters. The number 0 is the highest (earliest) grade; z is the lowest (latest). The default is **N**.

-j Displays the job identification number of the process that is running the command on the specified system. Use this job ID with the BNU command **uustat** to check the status of the command, or with **uustat -k** to terminate the process.

-n Prevents user notification by **mail** of the success or failure of a command. The default is to notify the user if the command fails.

-p Uses the standard input to **uux** as the standard input to *cmdstring*. A - (minus) has the same effect.

-r Prevents the starting of the spooling program that transfers files between systems. The default is to start the spooling program.

-s*file* Reports the status of the transfer in a file specified by *file* on the designated system.

Note: File names can contain only ASCII characters.

-x*debug_level* Displays debugging information on the screen of the local system. The *debug_level* is a number between 0 and 9. The higher number gives a more detailed report.

-z Notifies the user if the command completes successfully. This flag is the opposite of the system default, which is to notify the user of failure.

Examples

1. To get the *jobid* of a job and then compare a file on the local system zeus with a file on a remote system when the **diff** command is stored on the local system, use either of the following formats:

   ```
   uux -j "/bin/diff /usr/amy/f1 hera!/u/amy/f2 > ~uucp/f1.diff"
   ```

   or

   ```
   uux -j /bin/diff /usr/amy/f1 hera!/u/amy/f2 \{~uucp/f1/diff\}
   ```

   This command gets the file /usr/amy/f1 from the remote system hera, compares it to the file /u/amy/f2 on the local system (zeus), and places the output of the command in the local public directory in a file named f1.diff. (The full path name of this file is /usr/spool/uucppublic/f1.diff.) Using the **-j** option produces the output zeusN52d9.

   Note: As shown in the example, the destination name must be entered either preceded by a > with the whole command string enclosed in " . . . ", or entered enclosed in braces and backslashes, as \{ . . . \}.

2. To compare files that are located on two different remote systems, hera and venus, using the **diff** command on the local system:

   ```
   uux "!/bin/diff hera!/usr/amy/f1 venus!/u/amy/f2 > !f1.diff"
   ```

   This command gets the /usr/amy/f1 file from the system hera and the /u/amy/f2 file from venus, runs a **diff** command on the two files, and places the results in the file f1.diff, located in the current working directory on the local system.

   - This output file must be write-enabled. If you are uncertain about the permission status of a specific target output file, direct the results to the public directory, as in the first example.
   - The exclamation points representing the local system are optional.
   - Both of the examples above use a > symbol preceding the name of the output file. When using the special shell characters >, <, ;, or |, either put the entire *cmdstring* in quotation marks, or put the special characters as individual arguments in quotation marks.

3. To specify an output file on a different remote system:

   ```
   uux hera!uucp venus!/u/amy/f1 \{merlin!/u/geo/test\}
   ```

   This command runs **uucp** on system hera. The **uucp** command then sends the file /u/amy/f1, stored on system venus, to user geo on system merlin as test.

4. To get selected fields from a file on system `hera` and place them in a file on the local system:

   ```
   uux "cut -f1 -d: hera\!/etc/passwd > ~uucp/passw.cut"
   ```

   This command runs **cut** on the local system, gets the first field from each line of the password file on system `hera`, and places the output in the file `passw.cut` in the public directory on the local system.

   Note: In this example, **uux** is running in a c shell, so a \ (backslash) must precede the exclamation point in the name of the remote system.

Files

/usr/spool/uucp	Spooling directory.
/usr/lib/uucp	Contains the **uucico** daemon.

Related Information

The following commands: "**mail, Mail**" on page 608, "**uucico**" on page 1139, "**uucp**" on page 1144, "**uustat**" on page 1158, and "**uuxqt**" on page 1172.

uuxqt

Purpose

Executes remote command requests.

Syntax

AJ2FL117

Description

The Basic Networking Utilities (BNU) program **uuxqt** executes specified commands on designated remote systems.

Once **uux** is entered by a user, the program creates the necessary command (**C.***), data (**D.**), and execute (**X.***) files and places them in the spooling directory on the designated system. The **uux** command then calls the **uucico** daemon, which in turn attempts to contact the designated system to deliver the files. When the files have been transferred, **uuxqt** executes the commands on the designated system.

The **uuxqt** program searches the spool directories on the designated system for **X.*** files whose names indicate that they have been sent from another system. The command checks each execute file to make sure that:

- All the required data (**D.***) files are available and accessible
- File commands are permitted for the requesting system.

Note: BNU uses the **Permissions** file to validate file accessibility and command execution permission.

The **uuxqt** program, one of the Basic Networking Utilities (BNU) daemons (a program executed to handle file transfers and command executions), can be executed manually by an individual with superuser privileges. This daemon is executed automatically by the **uudemon.hour** shell script, which is started periodically by **cron**.

Flags

-s*system*	The name of the remote system. Use only when starting **uuxqt** manually. The *system* name is supplied internally when **uuxqt** is started automatically. **Note:** System names can contain only ASCII characters.
-x*debug_level*	Displays debugging information on the screen of the local system. The *debugg_level* is a single digit between 0 and 9. The higher the number, the more detailed the debugging information.

Files

/etc/locks/LCK*	Prevents multiple use of device.
/usr/adm/uucp/Maxuuxqts	Limits remote command executions.
/usr/adm/uucp/Permissions	Access permission codes.
/usr/spool/uucp/*	Spooling directory.

Related Information

The following commands: "**uucico**" on page 1139, "**uucp**" on page 1144, "**uustat**" on page 1158, and "**uux**" on page 1166.